型番:USBH-ACS30　　REV.2.0.0

USBメモリー
簡単シリアルアクセスボード

取扱説明書
お使いになる前にこの説明書をよくお読みの上正しくお使いください。本製品のサポートは製品の開発元である米GHI Electronics社が行います。

(C)2014 マイクロテクニカ

はじめに本マニュアルについて

この度は、USBメモリー簡単シリアルアクセスボード(型式USBH-ACS30、以下型式で記載)をご利用頂き、誠にありがとうございます。

本マニュアルは、本製品の開発元である米GHI Electronics社のマニュアルを元に、当方が日本語訳や解説等を追加して作成された日本語マニュアルです。英語版マニュアルではわかりにくい部分を図解で解説するなどしてマニュアルとして読みやすく編集しておりますが、すべての部分において完全な日本語訳となっていない部分がございます。

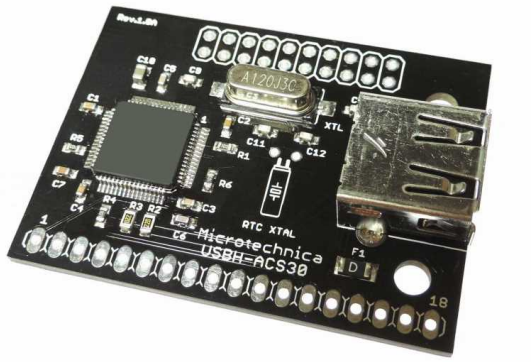
翻訳に誤りが存在する場合もございますので、開発元よりリリースされている最新の英語版マニュアルを必ずご参照頂き、この日本語マニュアルと併せてお読み頂けますようお願い申し上げます。

■開発元のオリジナルマニュアルダウンロード先
http://www.ghielectronics.com/downloads/ALFAT/ALFAT%20SoC%20Processor%20User%20Manual.pdf

※当方のマニュアルダウンロードページ(http://www.microtechnica.net/)からもダウンロードできます。

なお、日本語マニュアル作成においては適正かつ正確なものとなるよう注意を払っておりますが、場合によっては正当性や妥当性に不確実な部分が含まれていることもございます。日本語マニュアルに記載されている情報の利用による損害が発生しても当方では責任を負いかねますので予めご了承下さい。

製品本体



製品の概要

USBメモリー簡単シリアルアクセスボード(型式:USBH-ACS30、以下型式で記載)は、USBメモリーに簡単なシリアル通信コマンドで、アクセスすることのできるボードです。USBメモリーを使うためにUSBホストコントローラを内蔵し、FATファイルシステムを搭載しています。

USBH-ACS30はマイクロソフト社のロングファイルネーム命名(LFN規則)のライセンスを取得しており、FAT16、FAT32のファイルシステムに準拠した文字列が長い名前のファイル操作ができます。

ファイルの新規作成や更新、ディレクトリの作成、ファイルやディレクトリの削除などファイル操作を簡単に行うことができます。ファイルシステムはWindowsと同じFAT形式ですので、USBH-ACS30で作成、更新したファイルはそのままWindowsで扱うことができます。

RTC内蔵なので、ファイルにタイムスタンプをつけることができます。また付属の32.768KHzクリスタル発振子を取り付け、外部にコイン電池などを取り付けることで、バッテリーバックアップにてRTCの時間計測をメイン電源切断後も維持できます。

※本製品は、従来品のUSBH-ACS20と基板サイズ及びピンアサインが同一で物理的な置き換えが可能です。但し、コマンドは互換性がありませんので制御にはご注意下さい。

パッケージの内容

■同梱物
・USBH-ACS30本体
・32.768KHzクリスタル発振子
・マニュアル(本書)　※マニュアルはダウンロードにて提供

対応できるUSBメモリーとUSBメモリーの脱着

USBH-ACS30にはUSB2.0及び3.0規格のUSBメモリーが装着できます。速度はフルスピードに対応しています。対応するファイルシステムは、Windows標準のFAT16及びFAT32です。NTFSには対応しておりません。

使用可能なメモリー容量は512MB～64GBまで可能です。但し、サイズが大きくなるに伴い、初期化コマンドの完了に時間がかかるようになります。不必要に大きなサイズのUSBメモリーを使うことは推奨されません。

暗号化機能のついたUSBメモリー、ウイルス駆除機能のついたUSBメモリーなど特殊機能が付加されたものは使用できません。またメーカーが独自に付加した高速転送機能などは使えない他、場合によっては相性問題が発生する場合があります。

■USBメモリーの装着/取り外し
USBメモリーは、USBH-ACS30基板にあるUSBタイプAコネクタに装着します。USBメモリーの脱着は電源通電中でも可能です。

アクセス速度について

USBH-ACS30で使用されている米GHI社製のALFAT-SoCは、ARMコアのCortex-M3プロセッサを使用しており、高速なアクセス速度を実現しています。

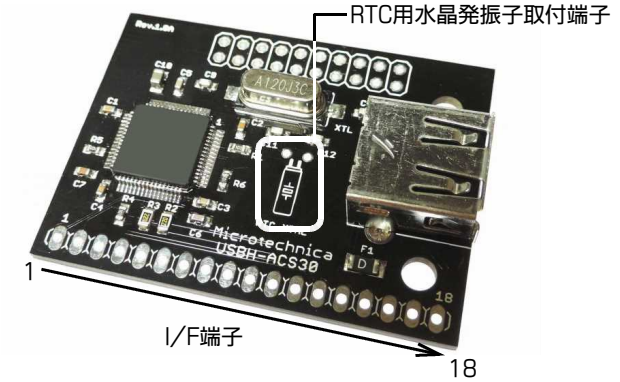
理論値で最高1000KBytes/secファイルアクセススピードを実現しています(USBフルスピード規格)。

ただし、この速度はあくまでも最高理論値であり実際にこの速度が達成できるわけではありません。本製品には実際にUSBメモリーにデータに書き込み、読み込みをして転送速度を計測するコマンドがありますので、そちらで実測値をお測りください。例として下記にUSB2.0のUSBメモリーとUSB3.0のものをそれぞれ使った場合の実測値を示します。サイズはどちらも8GBのUSBメモリーです。実測値ではUSB2.0/3.0規格での速度差はほとんどありません。

・USB3.0タイプ:書き込み→972.7kB/s　読み込み→878.4kB/s
・USB2.0タイプ:書き込み→939.3kB/s　読み込み→828.1kB/s

端子の概要とその詳細

■端子の概要



■I/F端子A

ピン	端子記号	内　容
1	UART TX	UART送信データ
2	UART RX	UART受信データ /I2Cビジー
3	I2C SCL	I2C通信クロック
4	I2C SDA	I2C通信データ
5	MODE1	モード設定ピン1
6	UART BUSY	UARTビジー
7	NC	
8	MODE2	モード設定ピン2
9	ACTIVE	アクティブピン
10	NC	
11	WAKE	ウェイクアップピン
12	Vbat	RTC電源及びVbatピン
13	Vcc	電源電圧+3.3V
14	$\overline{\text{RESET}}$	リセットピン(5Vトレラントではありません)
15	GND	電源GND
16	NC	
17	NC	
18	USBVcc	USBメモリー用電源 +5.0V

※NCは"No Connect"の略です。このピンには何も接続しないでください。

USBH-ACS30の電源電圧は3.3Vであり、ロジックレベルはLVTTLレベル(3.3V)ですが、すべてのI/OピンはTTLレベル(5V)のロジックもそのまま接続できます。(5Vトレラント機能搭載)

但し、14ピンのリセットピンは5Vトレラントではありません。+5Vをリセットピンに印加すると破損しますので十分ご注意ください。

■電源について

USBH-ACS30の電源電圧は本体動作用が+3.3V、USBメモリー駆動用が+5.0Vの2電源です。

消費電流は+3.3V側は待機時で平均30mA程度です。USBメモリー装着時で初期化コマンド実行時に約50mA～60mA程度の電流が流れます。+5.0V側のUSBメモリー用電源は使用するUSBメモリーに直接給電されUSBメモリーの種類などによって大きく変わります。一般的には最大で100mA程度の電流が流れます。

電流は動作状態や使用しているUSBメモリーなどによって変動しますので、+5.0V側は300mA程度は流せる余裕を持った電源をお使いください。+3.3V側は200mA程度は流せるものをご使用ください。また電源は必ず安定化されたものをお使いください。

Vbatピンは、RTC(リアルタイムクロック)用の電源ですが、RTCを使用しない場合でも、Vbatピンには+3.0V～+3.3Vの電圧を印加しておく必要があります。 ここに電圧が印加されていない場合RTCを使用していない場合でも本体が動作しませんのでご注意ください。

RTCを使う場合の電源接続例は次の"リアルタイムクロックの使用について"の項をご覧下さい。

> **?** RTCは何に使うの?
> RTCはファイルを作ったり更新したりする時にタイムスタンプを付けるとき必要になる日時管理をする機能です。本機ではRTCのクロック源として、ボードのCPUを動作させるクロックと共有させるモードと、基板に32.768KHzの水晶発振子を取り付けてバッテリーバックアップが可能なモードと2つあります。
> ちなみに、RTC機能を使わない(設定しないで)でファイルを作成したり更新したりすると、ファイルのタイムスタンプは意味のない日時になります。

■リアルタイムクロックの使用について

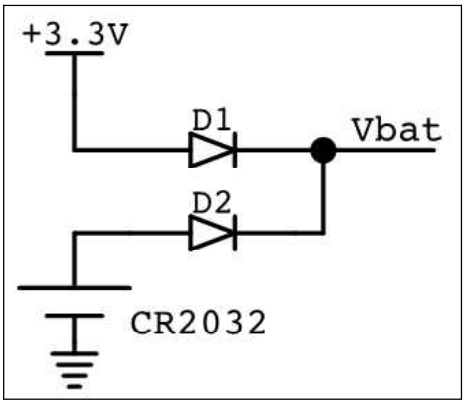
USBH-ACS30には、本体にリアルタイムクロック(RTC)を搭載しています。RTCは、時間を計測してファイルやフォルダの作成時に、タイムスタンプを付けるために使用されます。RTCの動作用クロックは2つから選択可能です。1つはCPU動作と同一のクロックを使う"シェアードモード"で、このモードの場合本体の電源を切断すると設定されていた日時データは失われます。そのかわり外付けの水晶発振子を取り付ける必要がありません。

もう1つのモードは"バックアップモード"で、基板の"RTC XTAL"に取り付けた32.768kHzの水晶発振子をクロックとして使います。このモードでは、Vbatピンに+1.65V～3.6Vの電源を給電し続けることで、RTCは時間計測を継続し日時データが本体の電源を切断しても維持されます。CR2032などのボタン電池でバックアップすることができます。

Vbatピンの消費電流は、3.3Vの時に約0.96μA程度～約1.68μA(平均値)でCR2032を使用した場合、公称容量が220mAhなので仮に消費電流を平均1.32μAとすると22655.5時間(2年213日程度)は電池の寿命が持つ計算になりますが、1.65Vを下回ると時計計測の維持が難しいので実際には1年程度が目安となります。

どちらのモードで使用するかは、"T"コマンドで設定を行い、その後、"S"コマンドを使用して日時を設定を行います。なお、一度RTCの設定がクリアーされてしまった場合やRTCの動作モードを指定しない場合、RTCの保持する値はまったく意味のない日時になります。(※決まった値にはなりません。)よって、RTCがクリアーされた後のファイルに保存されるタイムスタンプは全く意味のない日時となります。

バックアップ機能を使用する場合には、下図のようにダイオードを2本使用して電源を切り替える簡易回路を作ります。



※ダイオードは1S1588互換品などが使えます。
図:RTCバックアップ用電源(CR2032)追加回路例

※Vbatピンは、RTCを使用しない場合でもUSBH-ACS30を動作させるために+3V～+3.3Vの電圧電圧を印加する必要があります。

■モード設定

シリアル通信の方式をモード設定ピン(5、8ピン)で行うことができます。設定はUART通信モード、I2C通信モード、ブートローダモード(ファームウエアをアップデートするモードです)です。

必ず電源投入前にシリアル通信の方式をモード設定ピンで設定してから電源を投入して通信を開始してください。

通信モード	Mode1ピン(5ピン)	Mode2ピン(8ピン)
UART通信モード	0	0
I2C通信モード	0	1
ブートローダモード	1	0

※0はGNDへ接続、1は+3.3Vへ接続

■リセットピンについて

リセットピンは、GNDに接地するとハードウエアリセットがかかります。すべての設定はリセットされ初期状態に戻ります。(動作開始後のRTCのデータはリセットされません。)通常はタクトスイッチ等を接続して本体のリセットができるようにします。プルアップは必要ありません。

なおこのリセットピンは5Vトレラントではありません。絶対に+3.3V以上の電圧を印加しないでください。3.3V以上の電圧を印加すると本体はすぐに破損します。特にマイコンなどと接続してマイコンからリセットできるように設計する場合などは十分な注意が必要です。

■WAKEピンと省電力モードについて

WAKEピン(11ピン)は、"Z"コマンドによって省電力モードに移行した本体を通常状態に復帰させるためのピンです。

USBH-ACS30には次の2つの省電力モードがあります。

・スタンバイモード

システムを完全に停止させる省電力モードです。消費電流は10uA～25uA程度まで減少します。スタンバイモードに移行するとすべての設定はリセットされ、作業中の内容は破棄されます。ファイルハンドルも破棄され、ファイルハンドルのクローズ処理がされていないデータはすべて破棄されます。

復帰するには、リセットピンにより本体をリセットするか、WAKEピンに立ち上がりエッジパルスを入力します。

・ストップモード

現在の作業状態を維持したまま低消費電力状態になるモードです。消費電流は0.1mA～0.2mA程度になります。

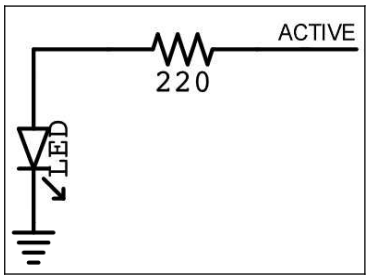
ストップモードから復帰すると、ストップモード移行直前の状態に戻ります。

復帰するには、WAKEピンに立ち上がりエッジパルスを入力します。

マイコンなどと、このWAKEピンを接続してコントロールする場合には、通常時はこのピンをLレベルにしておいて、省電力モードからの復帰時にこのピンに10ミリ秒以上のHパルスを印加します。システムは、パルスの立ち上がりエッジを検出して復帰します。

■アクティブピン(9ピン)について

アクティブピンは現在のコマンド実行状態を示すピンとして使います。通常はLEDなどを接続します。最大8mAの電流を流すことができます。USBH-ACS30がUART通信でデータを返す時にHレベルになります。コマンド送信が完了するとLレベルになります。LEDと接続する場合には、例えばVF=2VのLEDであれば220Ω程度の抵抗を介してGND間に接続します。(LED直結不可です)



但しこのピンはUART(非同期式シリアル)を使う場合には有効ですが、I2C通信モードで使う場合には使用できません。このピンはUART通信モードの場合のみ使用するようにしてください。

■UARTピン、I2Cピン等データピンの電気的特性

Symbol	Parameter		Conditions	Min	Typ	Max	Unit
V_{IL}	Input low level voltage		TTL ports 2.7 V ≤ V_{DD} ≤ 3.6 V	V_{SS}–0.3	-	0.8	V
V_{IH}(1)	TT(2) I/O input high level voltage			2.0	-	V_{DD}+0.3	
	FT(3) I/O input high level voltage			2.0	-	5.5	
V_{IL}	Input low level voltage		CMOS ports 1.8 V ≤ V_{DD} ≤ 3.6 V	V_{SS}–0.3	-	0.3V_{DD}	
V_{IH}(1)	TT I/O input high level voltage			0.7V_{DD}	-	3.6(4)	
	FT I/O input high level voltage				-	5.2(4)	
			CMOS ports 2.0 V ≤ V_{DD} ≤ 3.6 V		-	5.5(4)	
V_{hys}	I/O Schmitt trigger voltage hysteresis(5)			-	200	-	mV
	IO FT Schmitt trigger voltage hysteresis(5)			5% V_{DD}(4)	-	-	
I_{lkg}	I/O input leakage current (6)		V_{SS} ≤ V_{IN} ≤ V_{DD}	-	-	±1	µA
	I/O FT input leakage current (6)		V_{IN} = 5 V	-	-	3	
R_{PU}	Weak pull-up equivalent resistor(7)	All pins except for PA10 and PB12	V_{IN} = V_{SS}	30	40	50	kΩ
		PA10 and PB12		8	11	15	
R_{PD}	Weak pull-down equivalent resistor	All pins except for PA10 and PB12	V_{IN} = V_{DD}	30	40	50	
		PA10 and PB12		8	11	15	
C_{IO}(8)	I/O pin capacitance				5		pF

1. If V_{IH} maximum value cannot be respected, the injection current must be limited externally to $I_{INJ(PIN)}$ maximum value.
2. TT = 3.6 V tolerant.
3. FT = 5 V tolerant.
4. With a minimum of 100 mV.
5. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization, not tested in production.
6. Leakage could be higher than the maximum value, if negative current is injected on adjacent pins.
7. Pull-up and pull-down resistors are designed with a true resistance in series with a switchable PMOS/NMOS. This MOS/NMOS contribution to the series resistance is minimum (~10% order).
8. Guaranteed by design, not tested in production.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{OL}(2)	Output low level voltage for an I/O pin when 8 pins are sunk at same time	CMOS ports I_{IO} = +8 mA 2.7 V < V_{DD} < 3.6 V	-	0.4	V
V_{OH}(3)	Output high level voltage for an I/O pin when 8 pins are sourced at same time		V_{DD}–0.4	-	

シリアル通信について　～UART通信～

USBH-ACS30は非同期式シリアル通信(UART)と、同期式シリアル通信のI2Cに対応しており、モード設定ピン1(5ピン)と、モード設定ピン2(8ピン)にて、電源投入前に状態を設定をしてから起動します。

そのほか、本体のファームウエアをアップデートする際に使用するブートローダモードがあります。通信モードが正しくないと通信ができず動作しませんので、必ず5ピン、8ピンの設定を行った後に、電源を投入してください。

通信モード	Mode1ピン(5ピン)	Mode2ピン(8ピン)
UART通信モード	0	0
I2C通信モード	0	1
ブートローダモード	1	0

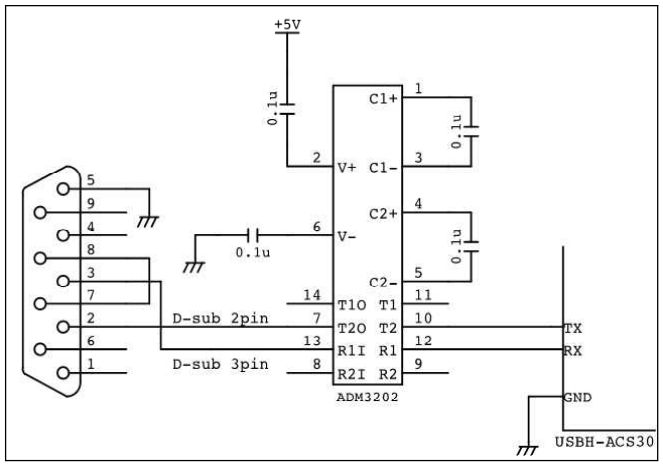
※0はGNDへ接続、1は+3.3V又は+5.0Vへ接続

■UARTモードでの使用方法

非同期式シリアル通信(UART)通信を行う場合にはマイコン又は、パソコンと接続する場合にはRS232Cポートを搭載したパソコンか、USB-UART変換IC(FT232R等)を介して接続します。

ロジックの電圧レベルは+3.3Vでも+5.0Vでも使用可能で、直結できます。下図はADM3202を使用してパソコンと接続するためのRS232Cインターフェイス(D-Sub9ピン)回路を作成した例です。

・パソコンとの接続回路例



USBH-ACS30の通信条件はデフォルト設定では下記の通りです。

・通信速度:　115200bps
・データ長:　8ビット長
・パリティ:　なし
・ストップビット:　1
・順序　LSBファースト

通信速度はシリアルコマンドで変更することができます。但し、設定した通信速度は電源を切断するとリセットされ、115200bpsに戻ります。通信速度を変更して使用する場合には必ず115200bpsで通信できる環境が必要となります。

■UART通信モードで使うピンについて

UART通信で制御する場合には次のピンを使います。

ピン名	ピン数	概　要
UART TX	1	データを出力するピンです(出力)
UART RX	2	データを受信するピンです(入力)
UART BUSY	6	USBH-ACS30がデータを受信できる状態かを通知する出力ピンです。このピンがHレベルの時はUSBH-ACS30はデータを受信できません。Lレベルになるまでデータを UARTホストから送ることは待機してください。このピンは常にUARTホスト側で監視しておくことをお奨めします。

1ピンと2ピンはUART信号の送受信ピンです。
USBH-ACS30ではRTS/CTSのフロー制御は搭載しておらず、その代わりにUART_BUSYピンがあります。このピンは、USBH-ACS30がデータを受信できる状態かを通知する出力ピンです。このピンがLレベルの時は、UARTデータを受信可能です。このピンがHレベルの時は、UARTデータは受信できません。Hレベルの時はLレベルになるのを待って、データを送信してください。2線式でTX/RX線だけを使って通信したい場合にはこのピンは使いませんが、データの量が多くなったり転送スピードが速くなるとデータの欠落が発生することがありますので、UARTホストではできるかぎりこのピンの状態を監視しながら通信をすることが推奨されます。フロー制御でいうところのRTS(Request To Send)ピンとほぼ同じ機能です。

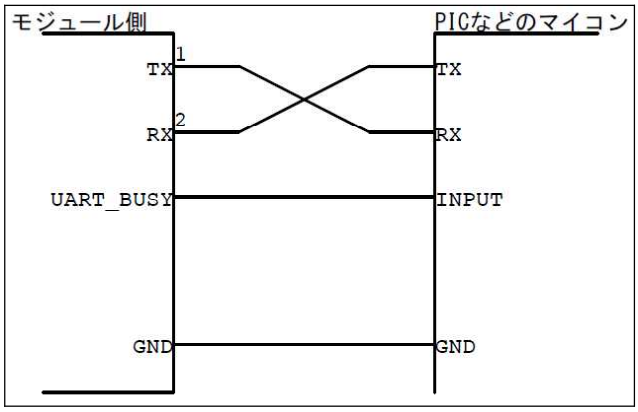
UART_BUSYピンを使わない場合にはオープンにしておきます。

■UARTの通信速度について

デフォルト(工場出荷時設定)では、115200bpsに設定されています。通信速度の変更は、115200bpsで通信を行い、Bコマンドによって通信速度を設定します。なお、設定した通信速度は電源を切断するとリセットされて115200bpsになります。(設定内容は記憶できません。)

■PICマイコンとの接続について

PICマイコンなどLVTTLレベル又はTTLレベルのマイコンと接続する場合、USBH-ACS30のTX及びRXピンはマイコンに直結できます。
ロジックレベルは0V-3.3Vですが、すべてのI/OピンはTTLレベル(5V)のロジックもそのまま接続できるようになっています(5Vトレラント)。よって、PICマイコンなど制御用マイコンがTTLレベルのデバイスであっても、そのまま直結できます。但し、14ピンのリセットピンは5Vトレラントではありません。+5Vを印加すると破損しますのでご注意ください。



上図はマイコンとの接続例です。

シリアル通信について　～I2C通信～

USBH-ACS30は、同期式シリアル通信のI2C通信でも制御が可能です。I2Cモードにするには、USBH-ACS30のモードピンを設定して、電源を投入します。

USBH-ACS30はスレーブデバイスとして動作します。スレーブアドレスは0xA4に固定されておりアドレス値は変更できません。また、1つのパスに複数のUSBH-ACS30を接続することはできません。

■USBH-ACS30をI2C通信モードに設定する

本体の電源を切断し、Mode1ピン(5ピン)をLレベル、Mode2ピン(8ピン)をHレベルと設定することでI2C通信モードになります。

設定は電源投入時にしか行えません。通電中のモード変更はできません。

通信モード	Mode1ピン(5ピン)	Mode2ピン(8ピン)
UART通信モード	0	0
I2C通信モード	0	1
ブートローダモード	1	0

■I2Cモード実行時のピンアサイン

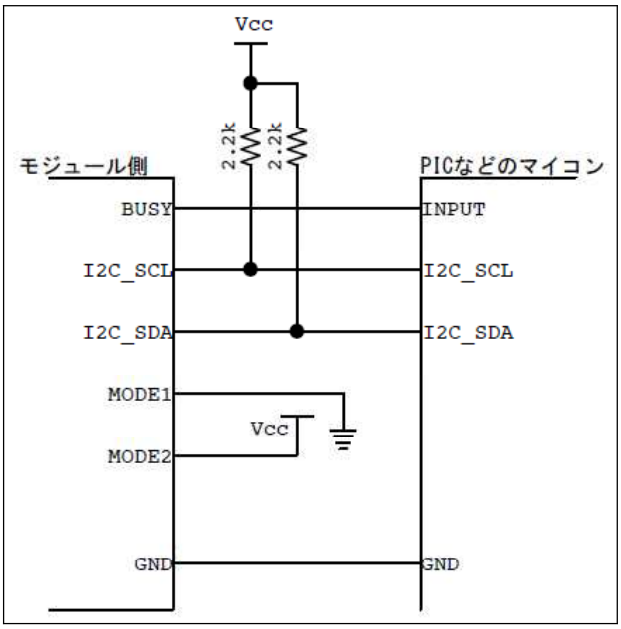
I2Cモードにすると、ピン配置は下記のようにアサインされます。

ピン	端子記号	内　容
2	BUSY	ビジー
3	I2C_SCL	I2C通信クロック
4	I2C_SDA	I2C通信データ

※その他のピンに変更はありません。

使用するピンは、クロックであるI2C_SCL(3ピン)と、データ線であるI2C_SDA(4ピン)です。また、BUSYピン(2ピン)を使用します。

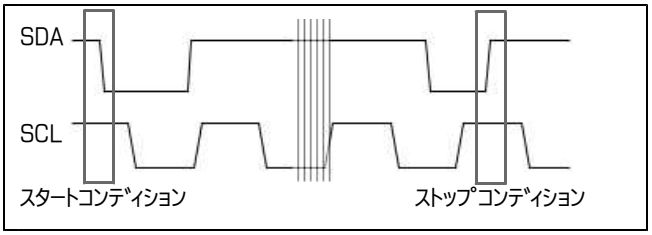
I2C_SCLと、I2C_SDAは共に2.2KΩ程度の抵抗でプルアップする必要があります。



BUSYピンはUSBH-ACS30の出力ピンで、I2Cマスター側から本ボードにデータを送信する間は、常に監視しておくことが推奨されます。このピンがHの時は、USBH-ACS30の内部入力バッファがいっぱいであることを示します。よってこの場合にはホストはデータを送信せずにBUSYピンがLレベルになるのを待つ必要があります。

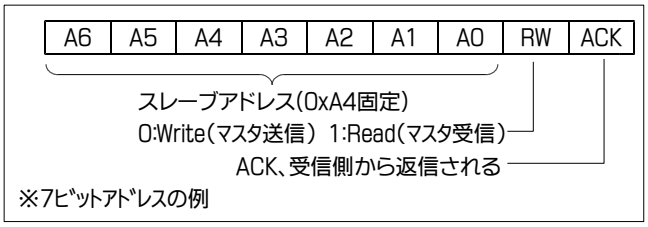
■I2C通信の決まり

マスタ側のSCL線がHレベルの時にSDAをLレベルにすると、スタートコンディションとなります。そのあと、続けてマスタがクロックの供給しつづけて最初の信号であるスレーブアドレスを送信します。なお、この際、スレーブアドレスのデータにRead/Writeを示す1ビットを付加してマスターが通信します。Readは1、Writeは0になります。USBH-ACS30のアドレスフォーマットは7ビットモードです。



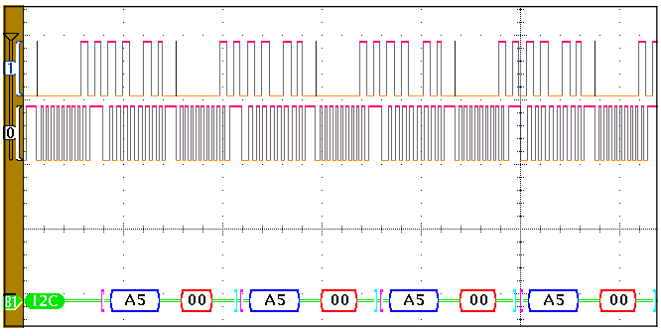
スレーブデバイスであるUSBH-ACS30は、マスタと指定されたR/Wの方向で通信を継続します。SCLのクロックに従い、送信側から8ビットのデータが出力されます。受信側ではこれに続いてACK信号を返信します。

最後のデータを送信し終わり、ACKを確認したあとにスレーブデバイス(USBH-ACS30)がSDAを解放するので、マスタ機器はSDA線をLレベルにしてクロックを停止し、SCLをHレベルにしてから、SDAをHレベルにすることでストップコンディションとなり通信は完了します。



USBH-ACS30はトークンを用いた方法でデータの送受信を行います。

USBH-ACS30からマスタ機器へ送るべきデータがない時、USBH-ACS30は、0x00を常に送信します。よって、マスタ側はこの0x00を受信する時はUSBH-ACS30側に送るべきデータがないことを把握できます。この0x00のことをNo Data Token(NDT)といいます。この状態をロジックアナライザで観察した波形が下図の通りです。なお、アドレスデータにReadモードの1を追加している関係でアドレス表示がA5になっていますが、実際は0xA4を指定しています。



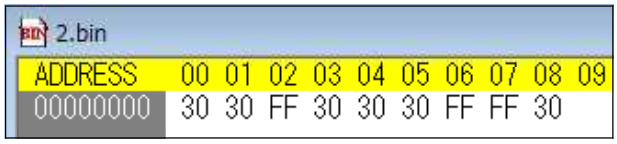
ただ、この方法の場合、もし送信したいデータの内容に0x00が含まれている場合、この0x00がNDTの0x00なのか、バイナリデータに含まれる0x00なのかを判断する必要があります。そのため、USBH-ACS30では、バイナリデータとして、0x00のデータを送信したい場合には、0xFFを送信し、続いて0x00を送信します。(2バイトになります。)この付加される0xFFをHalf Data Token(HDT)といいます。

さらに、もしバイナリデータが0xFFだった場合には、0xFFに続いてもう1つ0xFFを送信します。(2バイトになります。)まとめると次のようになります。

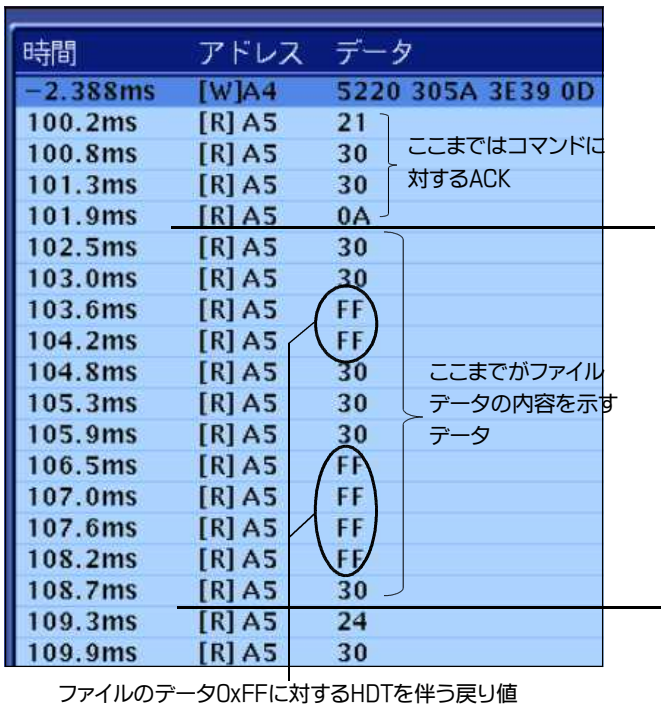
ファイル内の実データ値	USBH-ACS30から送られるデータ
0x00	0xFFに続いて0x00を送信
0xFF	0xFFに続いて0xFFを送信
0x01～0xFE	データのまま

よってこのことから、NDTとHDTが付加されるため、ファイルから読み込まれてUSBH-ACS30から送信されるデータは、実際のファイルサイズよりも大きくなることに注意してください。例えば、ファイル内のデータに0x00が含まれていればHDTの0xFFが付加され、0xFF,0x00の2バイトが送信されデータサイズが増えます。下記にそれ実例を示します。

例えば次のようなバイナリファイルをUSBメモリーに収録しておきます。

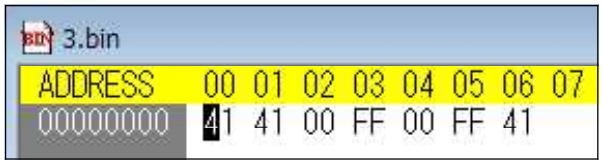


30hの間に0xFFが1カ所存在し、更にアドレス06と07に0xFFのデータが2つ続いています。このファイルをI2C通信で読み出すと次のようなデータがUSBH-ACS30から送信されます。下図はデータの内容をロジックアナライザで観察してデータを解析した例です。



ファイルのデータ0xFFに対するHDTを伴う戻り値

上図の通り、0xFFの実データに対してはHDTの0xFFが付加されていることがわかります。0xFFが2回続く場合、実際のデータは0xFFだとマスタ側の受信ソフトウエアで処理をするようにします。下記の例は、0x00を含む例です。



上のファイル例では、0x00と0xFFが交互に含まれています。このデータを含むファイルのデータをUSBH-ACS30てせ読み込んだ時のI2Cデータの内容をロジアナで観察してみます。その内容が次の図の通りです。

①の部分は実データでは0x00の部分です。先の説明通り0xFFに続き0x00が送信されています。また、実データが0xFFの所②では0xFFが2回続いていて実データが0xFFであることが分かります。
③も0x00であることがわかり、④は0xFFが2回続いていますので実データは0xFFであることがわかります。

0x00や0xFFが多く含まれるデータの場合にはマスタ側のソフトウエアでデータの内容を解析してデータを取り出すことが必要です。下図はC言語で記述した参考例です。

```
char Read_I2C() {

    Soft_I2C_Start();
    Soft_I2C_Write(0xA4+1);
    i2c_rec = Soft_I2C_Read(0);
    Soft_I2C_Stop();

    return i2c_rec;
}

void read_loop(){
    char b, data;

    do{
        b = Read_I2C();

        if((b!=0)&&(b!=0xFF)){
            data = b;
        }

        if(b==0xFF){
            b = Read_I2C();

            if((b!=0x00) && (b!=0xFF)){
                //ここは絶対に到達しない
            }else{
               switch (b){
                case 0x00:  data = 0; b=1, break;
                case 0xFF:  data = 0xFF;  break;
                }
            }
        }
    }while((b!=0));
```

上記は一例ですので必ずしもすべての環境で動作するわけではありませんが、参考例です。I2Cで読み込んだデータが0x00及び0xFFでない場合には、実データとしてそのまま扱います。(変数dataに値をそのまま代入しています。)

受信したデータが0xFFの場合には次に続くデータが0x00か、0xFFかをswitch文などで判定します。次に続くデータが0x00ならば、そのデータは0、もし0xFFが続いているようであれば、0xFFということになります。また受信はデータが0x00以外の時はdo～while文でループします。なお実データが0x00で、0xFF,0x00を受信した時、そのままですと、do～while文の判定でループを抜けてしまいますので、変数bに別の値を入れておきます。

コマンド体系等は、UARTとI2Cは共通です。

busy信号について

UART通信及びI2C通信におけるbusy信号は、本製品がデータを受け取れなくなった時にセットされる信号ですが、この信号は次の条件でセットされます。

・本製品内のバッファに蓄積されたデータが4kバイトを超えた時
・メモリーメディアにデータを書き込み及び読み込みしている時

本製品はシリアル信号を逐次バッファリングしながらメモリーメディアにデータを書き込んでいます。しかしながら、メモリーメディアに書き込む速度より、シリアル通信で送られるデータの方が速いときデータは本機のバッファに蓄積されていきます。
蓄積データのサイズが4kバイトを超える場合にbusyピンがセットされます。データがメモリーメディアに書き込まれ、バッファに空きができるとbusyピンは解除されます。

頻繁にbusyが発生する理由には次のような原因があります。

・1回に書き込むデータのサイズが大きすぎる
・メモリーメディアの書き込み、読み込み動作が遅い
・1つのディレクトリにファイルを作りすぎている
・メモリーメディアに頻繁にアクセスしすぎる
・メモリーメディアの容量が大きすぎる

特に同一のディレクトリにファイルを作りすぎている場合（500個以上など）、FATの構造及び仕様上、書き込み、読み込みに時間がかかるようになってきます。
ファイルの数は少ない方が望ましいですが、どうしてもファイルを多くしなくてはならない場合には、フォルダを作り各フォルダにファイル100個を上限として保存するようにすると改善が見込めます。

USBH-ACS30のシリアルコマンドの基本

USBH-ACS30のシリアルコマンドは次のような決まりがあります。

① USBH-ACS30はマイコンなどからUART通信又はI2C通信でシリアルコマンドを受け付けた後、必ずそれに対する応答を返します。
USBH-ACS30が正しくコマンドを受け付けた場合には、ACKが返ります。ACKは、"!00"という文字列です。(0x21,0x30,0x30,0x0A)最後の0x0Aは終端を表すラインフィードです。詳しくは③で説明しています。
一方、正しく処理ができない場合、エラーコードが必ず"!xx"の形式で戻ります。xxの部分には2文字の数字が入り、エラーの内容を示します。

このACK(!00)は、USBH-ACS30がコマンドを理解し正しくそれが実行されたことを示すものでエラーが無いことを通知するものです。またある処理が完了したことを通知する場合にも使われます。USBH-ACS30を制御するUARTホスト機器又はI2Cマスタ機器は、何かのコマンドをUSBH-ACS30に送信した時は必ずこの戻り値を確認するようにしなければなりません。このACKを確認した後、次の処理をするよう設計が必要です。ACKの確認なく次のコマンドを送信したり、別の処理を実行させると動作が正しく行われなくなり問題発生の原因となります。

② ホスト機器からUSBH-ACS30に送信するすべてのコマンドは、ASCIIコードの文字列コマンドです。例えば、本体のファームウエアバージョンを確認する"V"コマンドは、文字V(0x56)を送信します。

③ コマンドの最後は必ずラインフィード(LF=0x0A)で終端してください。先の"V"コマンドの場合には、文字VとLFの合計2バイト(0x56, 0x0Aを送信して初めてコマンドとして受け付けられます。ラインフィードで終端していない場合、コマンドとして実行されません。
終端はラインフィードだけでなく、キャリッジリターン(CR=0x0D)でも可能ですが、LFとCRの両方付けることは禁止されています。

④ USBH-ACS30から返されるすべての戻り値はASCIIコードの文字列です。ACKの"!00"も文字列です。また戻り値はすべてラインフィード(LF=0x0A)で終端されています。このLFを戻り値文字列の終端と認識するよう設計する必要があります。
但し戻り値は1行だけと限りませんのでUSBH-ACS30が送信データを送り終わるまで受信を続けてください。

【注意!!】
戻り値の終端はキャリッジリターン(0x0D)ではなく、ラインフィード(0x0A)なのでご注意ください。

⑤ いずれのコマンドも引数は200バイトを超えることはできません。これはコマンドのことであって、データの内容のサイズではありません。

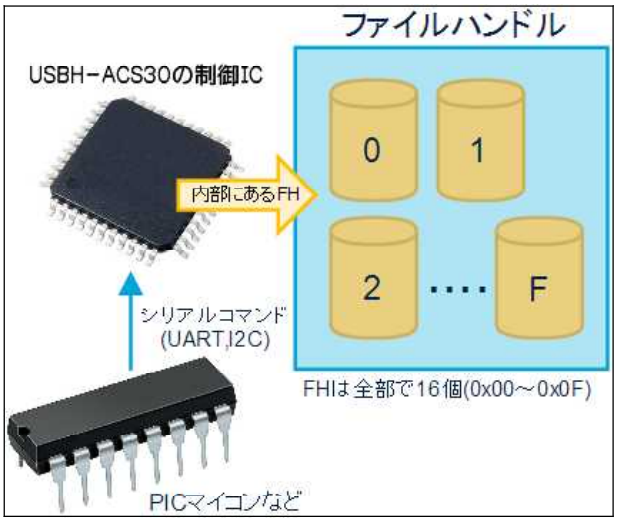
USBH-ACS30の基本的なファイル操作の仕組み

USBH-ACS30はファイル単位で操作しますが、ファイルは、新規に作成する時も、既存のファイルに追記する時も、そして既存ファイルからデータを読み込む時も、USBH-ACS30の内部にある"ファイルハンドル"(FHと記載されることがあります)と呼ぶ、バッファにファイルを開き、そのファイルハンドルに対して各種操作をします。

ファイルハンドルは16個(0x00～0x0F)あり、操作をしたいファイルを0コマンド(オーコマンド)でファイルハンドルに展開してから各種操作を行います。

0コマンドは、USBメモリーにファイルを新規作成する時も、既存のファイルに追記したり、ファイルからデータを読み込む時も使うコマンドです。指定したファイルハンドルにファイルを割り当てます。新規作成の時は、このファイルハンドルに空のファイルを作り、そこにデータを書き込んでいくというイメージです。

0コマンドでファイルを割り当てたら以後、そのファイルへのアクセスはファイル名ではなく、ファイルハンドルの番号で行うことになります。

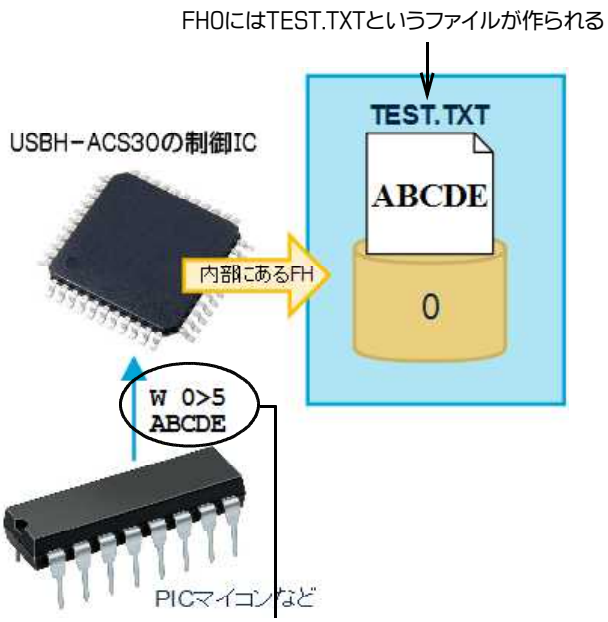


上図は概念図です。ファイルハンドルはファイルを一時的に開く箱のような物と考えると分かりやすくなります。

例えば簡単な例をご紹介します。新規に"TEST.TXT"というファイル名のファイルを作成し、そこに"ABCDE"という5バイトの文字列を書き込むことにします。

この場合には、0コマンドでファイルハンドル0(ゼロ)に"TEST.TXT"というファイルをWモードで割り当てます。ファイルハンドル0には、空の"TEST.TXT"というファイルが一時的に作られます。

以後は、ファイル名の"TEST.TXT"ではなく、ファイルハンドル0に対して各種操作をすることで、このファイルを自由に扱うことができるようになります。



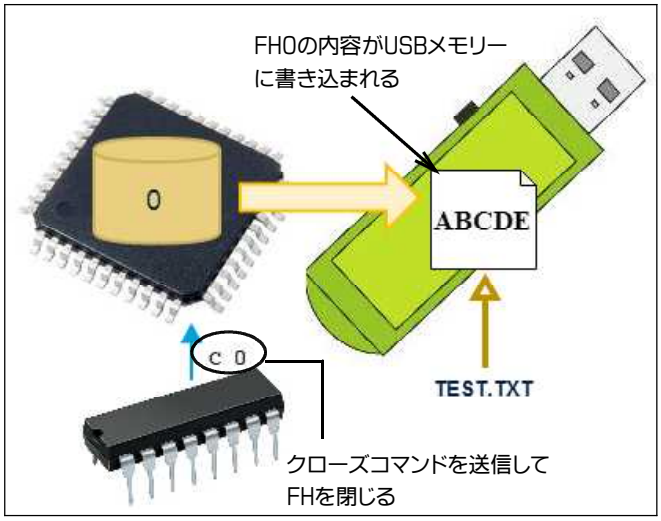
WコマンドでFH0に5バイトを書き込みます・・・というコマンドを送信後、書き込むデータ"ABCDE"を送信するとFH0内のTEST.TXTに"ABCDE"のデータが作られる。

上図は先の例の概念図です。これでファイルにデータを書き込むことができます。どんどん追記することもできます。

既存のデータからデータを読み込みたい場合でも同じ仕組みで行います。メモリーカードにあるファイルを指定して一度ファイルハンドルに開き、そこから読み出すという手順で操作します。

ファイルハンドルに作られたファイルは、一時的なファイルであるためこのままではUSBメモリーには書き込まれません。ここで何もせずに電源を切断したり、USBメモリーを本体から抜いてしまうと、ファイルハンドルに作られたデータはすべて破棄されてしまいます。ファイルハンドル内のデータをUSBメモリーに書き込むには、最後にファイルハンドルをクローズする処理が必要です。このクローズする処理をした時に、USBメモリー上にファイルが作られますので、ファイル操作を終了する場合には必ずクローズコマンドでファイルハンドルを閉じて終了します。

クローズコマンドでファイルハンドルを閉じてデータを書き込むと、そのファイルハンドルは空きになり、別のファイルを開くことができるようになります。



USBH-ACS30は先に説明の様に一度ファイルハンドルに開いたファイルの編集や読み込みについてはファイルハンドルを指定することで行いますが、その他の操作をする場合には、ファイル指定はフルディレクトリ名で指定します。USBメモリーは、U0: ドライブと決められています。例えばUSBメモリー内のファイル"TEST.TXT"を指定する場合には次のようになります。

U0:\TEST.TXT

なおU0の後ろは:(コロン=0x3A)です。また、コロンの後ろにあるディレクトリの区切りはバックスラッシュ(0x5C)です。

⚠ バックスラッシュ(5Ch)についてのご注意!!

バックスラッシュは、ASCIIコードで5Chですが、5Chを文字列のエスケープとして定めいているプログラミング言語(C言語など)では、文字列として1つ5Chを挿入してもエスケープ文字として処理され正しくコマンドが出力できません。

マイコンなどの開発をC言語で行っている場合、文字列として正しく引数を指定して文字列を出力したつもりでも、5Chが正しく出力されておらず引数が不正になる場合があります。

多くのC言語の場合、エスケープ文字である5Chを2つ重ねて記述することで、5Chそのものを表わすという仕様になっていますので、プログラム開発の際にはその点についてご使用言語の仕様をよくご確認頂けますようお願い致します。

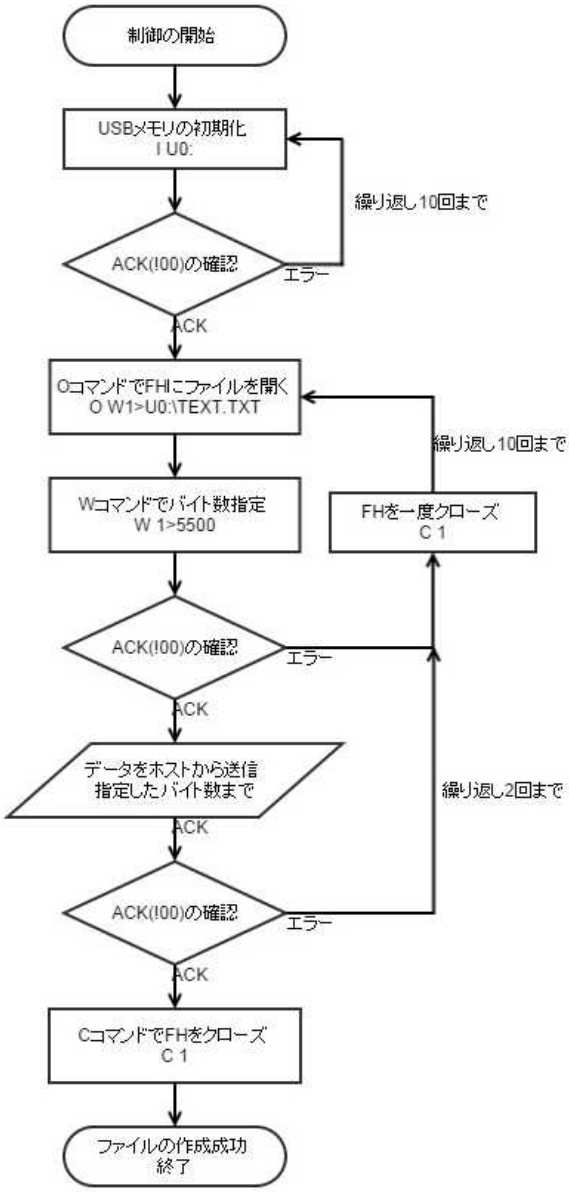
例:次のように記述すると、バックスラッシュが出力されずエラー(不正な引数)が通知される場合があります。

```
printf("O  1W>U0:\TEST.TXT\n")
                ↑
```

ここの5Chが送信されません。

これはいわゆるC言語の5C問題に該当します。"\\"として2回バックスラッシュを記述するなど対応が必要です。

USBH-ACS30で、USBメモリーに新規にファイルを作成する場合の制御例を下記に示します。



上図のフローチャートは最も一般的な制御方法例です。原則として各コマンドをUSBH-ACS30に送信した後には、それに対するACK(!00)を確認するようにします。ACKが返れば次の処理を実行、返らない場合にはエラーコードからエラーの内容を推測したり、再度処理を実行するなどしてエラーに対応します。ACKを確認しないで制御を進めていくと不正な動作をする場合があります。

USBH-ACS30の扱えるファイル形式と名前

USBH-ACS30の扱えるファイル形式とファイル名、ディレクトリ名には下記のような決まりがあります。

■扱えるファイル形式
FAT16及びFAT32
※ファイルアロケーションサイズは問いません

■ファイル名とディレクトリ名についての制限
USBH-ACS30では、ファイル名及びディレクトリ名にASCIIコードのみ扱えます。Unicodeなどの2バイト文字は扱えません。
次の記号は使用できます。
$ % ' - _ @ ! ( ) { } # &

■ファイル名とディレクトリ名の長さについて
USBH-ACS30では、ロングファイルネームに対応しています。
本製品はマイクロソフト社のFAT32におけるLFNのライセンスを受けており、長いファイル名が使用できます。
ファイル名は本製品からメモリーメディアのファイルにアクセスする場合には大文字小文字が区別されません。本製品でファイルを新規に作成する場合、ファイル名及び拡張子は必ず大文字になります。

USBH-ACS30で使えるUSBメモリー

USBH-ACS30では下記のメモリーメディアをサポートしています。

USBメモリー　USB2.0又はUSB3.0規格対応
512MB～64GBまで

※但しサイズが大きくなると初期化に時間がかかるようになります。また一部のコマンドで処理に時間がかかるようになります。できれば4GB～8GB程度のUSBメモリーをご使用ください。

なお、使用に際しては必ずメディアをパソコン等でFAT32形式でフルフォーマットしてから(クイックフォーマットではなく)ご使用ください。

USBH-ACS30の準備

USBH-ACS30を使用する際には、以下の手順でハードウエアの準備を行ってください。

1 あらかじめ使用するUSBメモリーをパソコンでFAT16又はFAT32形式でフォーマットしておきます。

2 USBH-ACS30本体にUSBメモリーを装着します。

3 USBH-ACS30のシリアル通信モードの設定を正しく行った後、電源を投入します。電源投入直後、下記のような文字列が返ります。

```
 GHI Electronics, LLC
 ----------------------
 ALFAT SoC Processor
!OO
```

※最初の"G"の前には、スペース(0x20)が1つ入っています。またその前(先頭)にはラインフィード(0x0A)が挿入されています。また一番最後はラインフィードで終端されています。

この電源投入時に返る文字列のことをバナーといいます。バナーは本体の電源を入れた時と、ハードウエアリセットをした時に必ず返ります。

4 これでハードウエアの準備が完了しました。

とりあえず使ってみましょう

USBH-ACS30には、さまざまな機能が搭載されており、シリアルコマンドも豊富です。しかし、ここではとりあえず動作させて使ってみることで、USBH-ACS30の最も基本的な使い方を知って頂くことにしましょう。詳しいシリアルコマンドについては次の項から記載がありますので、そちらでお読みください。

なお、本項ではUSBH-ACS30から返るコマンドは、斜体で記載します。また、コマンド内の<LF>はラインフィード(0x0A)、<sp>は半角スペース(0x20)を示します。USBH-ACS30からの戻り値は、すべてラインフィード(LF)で終端されているとして、<LF>は記載しておりません。

■USBメモリーに新規にファイルを作成する
まず基本的な操作であるファイルの新規作成を体験しましょう。USBメモリーをご用意ください。

1 USBメモリーをパソコンでフォーマットしてください。

2 USBH-ACS30の配線を行いましょう。今回はパソコンのRS232Cポートと接続して実験することにします。シリアルインターフェイスはUART(フロー制御は使用しません)を用います。
VccピンとVbatピン正しく電源電圧が、印加されているか確認しましょう。その他配線が正しく行われていることを確認しましょう。パソコンと接続する場合には、ADM3202等のレベル変換ICを介してパソコンのRS232Cポートと接続する必要があります。またモードピンはUARTモードに設定しておく必要があります。
UART通信のデフォルト通信速度は115.2kbpsです。

パソコン側はシリアルターミナルをご用意ください。当方ではフリーで使いやすいターミナルソフトをご用意しています。下記からダウンロードしてご使用ください。
http://www.microtechnica.tv/faq/faq.cgi?kate=Maeni&faq=4

3 USBH-ACS30にUSBメモリーを装着してください。

4 シリアル通信で、ラインフィード(0x0A)を1回送信してみてください。
USBH-ACS30が正しく動作しているとACKの!00が返ります。

```
<LF>
!00
```

ラインフィード(又はキャリッジリターン)を送信すると、USBH-ACS30が動作しているか、配線が正しいかを確認できます。

5 最初にUSBメモリーを初期化します。USBメモリーに対するすべての操作は必ず初期化コマンドを実行して、初期化が正しく完了した後から行えます。初期化コマンドは"I"(0x49)で、引数に初期化する対象ドライブを指定します。
ここではUSBメモリーなので、U0:を指定しますので次のようなコマンドを送信します。

```
I<sp>U0:<LF>
!00
```

USBメモリーが正しく初期化できると、ACKの!00が返ります。
必ず初期化を実行してACKが返ることを確認してから次の操作をするように設計してください。ここで初期化に失敗して!00以外のエラーコードが返った場合、そのメディアは使えません。

> メディアの初期化に失敗する場合、多くはエラー、!11が返ります。正しいドライブを指定しているか、メディアが正しく装着されているか確認してください。
> またメディアとの相性問題で正しく初期化できない場合もあります。

初期化は完了までに一番時間のかかるコマンドです。USBメモリーの種類、サイズ、フォーマット状態などによって変わりますが、5秒～50秒程度かかります。ACKが返るまでは初期化中ですので、一切の操作、コマンドの送信などはせずお待ち下さい。

6 初期化が正しく完了したらファイルを新規に作成しましょう。
ファイルハンドル1にファイルを作ります。ファイル名は"123.TXT"としてみましょう。FHに対してファイルを割り当てる場合には0コマンドを使います。パスはフルパスを指定します。

```
O<sp>1W>U0:\123.TXT<LF>
!00
```

ドライブ名の後ろにはコロン(0x3A)とバックスラッシュ(0x5C)が含まれることを注意してください。その後ろは>(0x3E)です。

引数の1はFHの値、Wはファイルの新規作成モードを指定する文字です。(読み込みはR、既存ファイルへの上書きはAです)

7 続いてFH1に何バイトのデータを書き込むか、書き込むデータサイズをWコマンドで指定します。指定するバイト数は16進数の値で指定します。例えば11バイトの場合には、"B"を指定します。

```
W<sp>1>B<LF>
!00
```

このWコマンドを送信後、ACKの!00がUSBH-ACS30から返ってきたところから、書き込むデータの受付が始まります。!00が返るまではデータを送信しても書き込まれませんので!00を必ず待つようにします。

サイズは11バイトを指定しました。コマンドの時はラインフィードで終端しましたが、書き込みデータの場合には終端は必要ありません。ファイルに書き込むデータだけを送信してください。
USBH-ACS30は11バイトのデータが到来するまでずっと待機を続けます。途中で待機を抜ける方法はありません。

※指定したデータサイズよりも、書き込むデータサイズが小さい場合には不足分をラインフィードやキャリッジリターンのデータを埋め合わせて書き込み待機を終了させる方法もあります。

8 ここでは11バイト、すなわち11文字の文字列"HELLO WORLD"ファイルに書き込んでみましょう。

```
HELLO WORLD
$0000000B
!00
```

書き込むデータが指定したデータサイズに達すると、自動的にUSBH-ACS30から、書き込みが完了した旨の通知と、ACKが返ります。

"*$0000000B*"は、11バイトを確かに受信しました、という通知です。その次の!00は書き込み完了のACKです。
このACKで書き込み完了を確認してください。

9 ここまでの作業でデータはUSBメモリーのメモリー領域に書き込まれました。しかしFATファイルシステムでは、データを書き込んだだけではそのデータを使うことはできません。
データの存在する位置を管理領域というテーブルに書き込んで初めて「使えるデータ」となります。

USBH-ACS30では、この作業を行うためにクローズコマンドの"C"コマンドを使います。Cコマンドを使用すると、指定したファイルハンドルが閉じられ、データが参照できるようになります。
コマンドを次のように送ります。

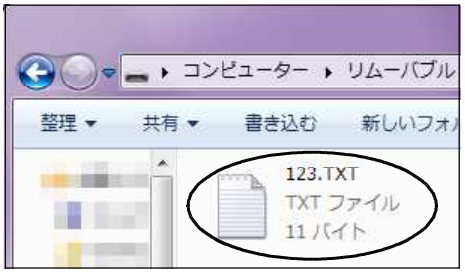
```
C<sp>1<LF>
!00
```

ファイルハンドル1を閉じるというコマンドです。

ACKの!00が返った時点で、USBメモリーには"123.TXT"というファイルが作成されます。

※このCコマンドを送信しないで、USBメモリーを抜いてしまったり、電源を切断してしまったりすると、データはすべて消えてしまいますので十分ご注意ください。

10 ではこのUSBメモリーをパソコンに装着して、正しくファイルが作成できているか確認しましょう。



ファイルが作られていることが確認できます。
メモ帳で開いてみましょう。

正しく"HELLO WORLD"という文字列が書き込まれていることが確認できます。

このようにUSBH-ACS30ではシリアルコマンドで簡単にファイルを作成することができます。もちろんテキストファイルだけでなく、形式を問わず作成できます。

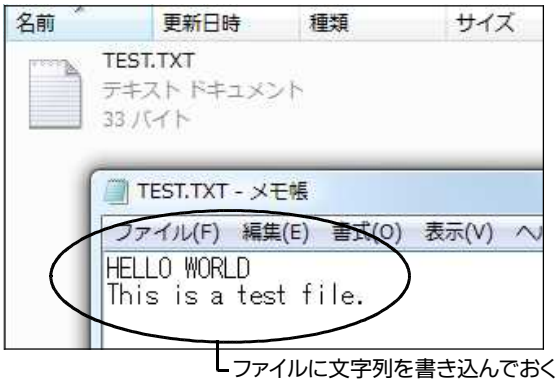
ここでは簡単なファイル作成例を紹介しましたが、様々なシリアルコマンドを使うことで、さらに複雑な操作ができます。

続いてファイルからデータを読み出す方法も紹介します。

■テキストファイルからデータを読み込む

基本的な操作である、USBメモリー内のテキストファイルからデータを読み出す操作を体験しましょう。先の書き込む例と同じようにUSBメモリーを使います。

1 データを読み出すためにあらかじめ、USBメモリー内にファイルを保存しておきます。
"TEST.TXT"という名前のファイルに次のような文字列を記述してUSBメモリーのルートディレクトリに保存しておきます。



ファイルに文字列を書き込んでおく

2 先のファイル作成の例と同じで、まずUSBメモリーを本体に装着し、"I"コマンドで初期化を行います。

```
I<sp>U0:<LF>
!00
```

3 今回は、データを読み込むモード(Rモード)で、ファイルハンドルにファイルを展開しますので、次のようにコマンドを送信します。

```
O<sp>1R>U0:\TEST.TXT<LF>
!00
```

ACKが返ったことを確認します。

4 読み出しはRコマンドで行います。
読み出すデータのサイズを指定します。指定は16進の値で行います。ここでは試しに16バイト(0x10)を指定してみましょう。

```
R<sp>1Z>10<LF>
```

FH指定の数字1の後ろにある"Z"(ゼット)は、超過文字でファイルハンドル内にあるファイルのデータサイズが、指定したデータサイズよりも小さい場合、超過分のデータをこの文字で埋めるというものです。データサイズ内であれば、特に気にする必要はありません。

5 上記のコマンドを送信すると、下記のようにデータが返ります。

```
!00
HELLO WORLD
Thi$00000010
!00
```

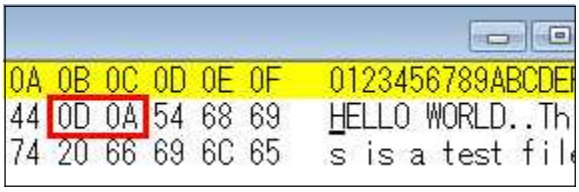
最初の!00はACKで、その後から実際のデータが返ります。データの最後には$(0x24)が付き、読み取ったデータサイズ、その次に再度読み取り完了のACK、!00が返ります。

よってホスト機器側では、最初のACK、!00の後ろにあるラインフィード(0x0A)から、$(0x24)までの区間にあるデータがファイルから読み取ったデータであると判別するようにします。

ところで、ここでは読み取るデータサイズの指定を16バイトとしました。しかし文字数を数えると14バイトしか返ってきていないように見受けられます。

これは間違えではありません。
なぜならば、文字列"HELLO WORLD"と"This is a test file"の文字列の間には改行が入っているからです。

Windowsシステムでは、改行はキャリッジリターン(0x0D)と、ラインフィード(0x0A)の2バイトから構成されます。よって、Windowsでテキストデータを作った場合、改行の部分は2バイトのデータとして扱われます。
先ほどのTEST.TXTをバイナリエディタで見てみましょう。



改行部分に0x0Dと0x0Aが含まれていることが分かります。この2バイト分が、読み込みデータの指定サイズに含まれているわけです。キャリッジリターンやラインフィードはテキストデータとして扱っていると意識しないことが多いので注意してください。

6 さて、続いて残りの部分も読み込んでみましょう。
今度はテストとして再度20バイトを指定してみましょう。

```
R<sp>1Z>14<LF>
```

返ってきた文字列は下記のようになりました。

```
!00
s is a test file.ZZZ$00000011
!00
```

Rコマンドでは、ポインタの位置をリセットしたり、任意の位置にセットしない場合、前回の続きの位置からデータを読み出します。(ポ

インタの位置指定はPコマンドで行えます。）

ここでは20バイトを指定しました。実際にはファイルにデータが残り17バイトしかありませんでしたので、超過した分の3バイトの部分に、超過文字"Z"が付加されました。

シリアルコマンド

■コマンドについての一般的な規則

USBH-ACS30を制御するためにはシリアルコマンドのやりとりをする必要がありますが、次の項目を守るようにシステムを設計してください。

・いずれのコマンドでも引数のサイズが200バイトを超えないようにしてください。

・各コマンドは必ずラインフィード(0x0A)で終端してください。 ※キャリッジリターン(0x0D)でも代用可能です。

・USBH-ACS30から戻る、戻り値を必ず確認するようにシステムを設計してください。
各コマンド送信後に戻り値をチェックできる仕組みを作り、その内容は必ず解析される必要があります。特にACKである"!00"なのか、それ以外のエラーコードなのかは最低限確認するようにします。
また次のコマンドを送信する際やデータを送る場合には、必ずACK確認後に行うように設計をしてください。

・扱うすべての番号はASCIIで表記された(文字)16進数の値です。例えば、10進数で数値10と指定したい場合には、文字"A"(0x41)を送信します。
誤って、10進数表現のまま"10"という文字列を送信してしまうと、USBH-ACS30は、16進数の"イチゼロ"として解釈します。
例: W 1>C →FH1に12バイト(0x0Cバイト)のデータを書き込む

・以降のUSBH-ACS30の制御コマンドを解説では、下記のような決まりで記述しています。
- *{LF}* はラインフィード(0x0A)を表します。 ※戻り値は終端のLFは省略しています。
- △印 はシングルスペース(0x20)を表します。
- 下記表記の中で斜体文字は、USBH-ACS30からの戻り値を表します。

■コマンド一覧表

コマンド	詳細	コマンド	詳細
V	ファームウエアバージョンの取得	I	メモリメディアの初期化
Z	省電力設定	O	空きのファイルハンドルにファイルを開く(割り当てる)
T	RTCの初期化	W	ファイルにデータを書き込む
S	現在時刻、日にちの設定	R	ファイルからデータを読み込む
G	現在時刻、日にちの取得	F	ファイルハンドルのデータをフラッシュする
B	UART通信速度の変更	C	ファイルハンドルを閉じる
#	エコーの設定	P	ファイル内のポインタ位置を移動
A	ファイルの名前変更	Y	ファイル内のポインタ位置を取得
E	メモリメディアのテスト	D	ファイル又はディレクトリの削除
K	メモリメディアの空き容量取得	?	ファイル又はフォルダーの存在の検索
@	ディレクトリリストの初期化	M	ファイルの内容を別のファイルにコピー
N	次のディレクトリエントリを取得	Q	クイックフォーマットの実行
J	USBメモリーの状態取得		

■各コマンドーの詳細

【コマンド】 **V***{LF}*

【動作】 ファームウエアバージョンを取得する

【戻り値】 *v2.0.0* バージョン値
!00

【コマンド】 **#**△ *n{LF}*

【動作】 エコーバック(受信したシリアルデータを、そのまま送り返す)の有効/無効を設定する

【引数】 n 0の時はエコーバック無効、 1の時はエコーバック有効

【解説】 エコーバックとは、USBH-ACS30が受信したシリアルデータをそのまま送信元へ返す処理です。
デフォルト設定では無効になっています。

【コマンド】 **Z** *△n{LF}*

【動作】 省電力モードの設定を行う、省電力モードに移行する

【引数】 *n* 省電力モードの番号を指定
0･･･スタンバイモード CPU消費電流平均約75uA
1･･･ストップモード CPU消費電流平均約350uA

【戻り値】 なし

【解説】 2つの省電力モードがあります。本コマンドを実行するとすぐにモードが変更されます。

■スタンバイモード
システムを完全に停止させる省電力モードです。消費電流は20uA程度まで減少します。
スタンバイモードに移行するとすべての設定はリセットされ、作業中の内容は破棄されます。ファイルハンドルも破棄され、ファイルハンドルのクローズ処理がされていないデータはすべて破棄されます。
復帰するには、リセットピンにより本体をリセットするか、WAKEピンに立ち上がりエッジパルスを入力します。

■ストップモード
現在の作業状態を維持したまま低消費電力状態になるモードです。消費電流は0.1mA～0.2mA程度になります。ストップモードから復帰すると、ストップモード移行直前の状態に戻ります。
復帰するには、WAKEピンに立ち上がりエッジパルスを入力します。

マイコンなどとこのWAKEピンを接続してコントロールする場合には、通常時はこのピンをLレベルにしておいて、省電力モードからの復帰時にこのピンに10ミリ秒以上のHパルスを印加します。システムは、パルスの立ち上がりエッジを検出して復帰します。

【補足】 スタンバイモードから、WAKEピンへの立ち上がりエッジ信号印加により復帰した場合には、電源投入時及びハードウエアリセット時と同じ次のバナーが返ります。

```
 GHI Electronics, LLC
----------------------
 ALFAT SoC Processor
!00
```

ストップモードからWAKEピンへの立ち上がりエッジ信号印加により復帰した場合には、ACKの"!00"が返ります。
本コマンドで省電力モードに本体が移行しても、USBメモリーの消費電流に変化はほとんどありません。

【コマンド】 **T** *△mode {LF}*

【動作】 リアルタイムクロックのクロック源選択と、初期化を行う

【引数】 *mode* RTCに使うクロック源の選択
S･･･シェアードモード: RTC用クロックをCPUのクロックと共有する(バックアップ不可)
B･･･バックアップモード: 基板上X5に取り付けた32.768kHzの水晶発振子をクロック源として使用する(バックアップ可)

【戻り値】 *!00*

【解説】 ファイルを作成する際などにタイムスタンプとして使用するRTCの初期化と、クロック源の指定を行います。
クロック源は2つあり、1つはCPUを動かしているクロックを使用する方法でシェアードモードといいます。このモードの場合外部に水晶発振子を取り付ける必要はありませんが、日時データのバックアップはできません。(VBATピンに電源を印加し続けても日時データは保持されません。)

もう1つのモードは、基板上の"RTC XTAL"に32.768kHzの水晶発振子を取り付けて、これをRTC用のクロック源として使う方法です。このモードの場合にはVBATピンに1.65V～3.6Vの電源を、本体の電源切断後も供給し続ければ日時データは保持されます。

RTC機能を使用するためには、最初にTコマンドで初期化及びクロック源を設定後、現在日時設定コマンド(Sコマンド)で日時を設定します。Sコマンドで現在日時を受信した段階から、時計計測が開始されます。

【補足】　RTCの日時データは設定しないと、意味を持たない値となります。
RTCを使わなくてもファイル、ディレクトリの作成やデータの読み取りそのものには不具合は生じません。
本機のRTCは2秒毎の精度です。(奇数秒は設定及び取得、タイムスタンプとしての書き込みはでません。)

【コマンド】 **S** *△ ddddtttt{LF}*

【動作】　現在日時をDWORD形式で設定する （本コマンド実行前にTコマンドの実行が必要です）

【引数】　*ddddtttt*　ddddは、年月日を16ビット長で、ttttは時分秒を16ビット長でそれぞれ表します。

【解説】　内蔵のRTCに日時を設定します。
引数のddddtttttは、DWORD型(32ビット)で現在日と現在時間を設定します。ビットの割り当ては下記の通りになっています。
下表の例では、2014年4月1日　18時30分00秒　と設定する場合についての値が書かれています。

ビット	内容	詳　細	例
31～25	年	現在の年から1980を引いた差	2014−1980=34 →2進数で 0100010 （7ビット長）
24～21	月	1月～12月	4月　→2進数で 0100 （4ビット長）
20～16	日	1日～31日	1日　→2進数で 00001 （5ビット長）
15～11	時	0時～23時	18時　→2進数で 10010 （5ビット長）
10～5	分	0分～59分	30分　→2進数で 011110 （6ビット長）
4～0	秒	秒を2で割った数　0～30	00秒　→2進数で 00000 （5ビット長）

例で2進数に換算した値を並べると、32ビットになるので、"01000100100000011001001111000000"となります。
これを16進数に換算すると、"448193C0"となりますので、"S 448193C0"として送信し現在日時を設定します。

このSコマンドで設定した日時を、USBH−ACS30が受信した時点から、設定日時からの時計計測が開始されます。
"S"コマンド送信の前に必ずTコマンドを送信してRTC機能を初期化させておきます。

※FATでは32ビットの時間管理が標準となっていますので、USBH−ACS30もこの方法に従っています。

【コマンド】 **G** *△ mode{LF}*

【動作】　RTCから現在の時間又は日にちを取得する

【引数】　*mode*　時間か日にちを指定します
D･･･現在の日にち
T･･･現在の時間

【戻り値】

日にちを指定した場合	時間を指定した場合
MM−DD−YYYY	*HH:MM:SS*
!00	*!00*

【コマンド】 **B** *△ssssssss{LF}*

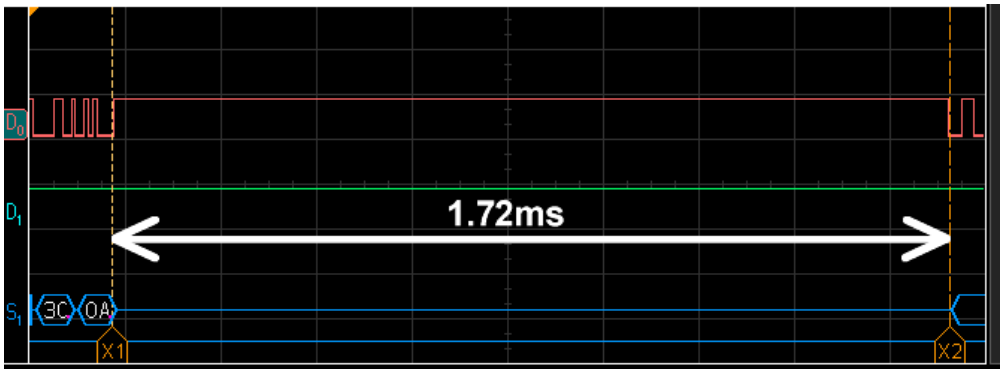
【動作】　UARTの通信速度を設定 （電源投入時、リセット後のデフォルト値は115.2kbps）

【引数】　*ssssssss*　設定したい通信速度をbps単位の16進表現の文字列で指定　※1200の倍数が望ましい

【解説】　UART通信の通信速度を変更します。デフォルト設定は115.2kbpsです。引数で指定する通信速度は16進で表現した文字列です。
例えば9600bpsに設定したい場合には、"B 2580" になります。(9600=0x2580)

Bコマンドを送信すると、!00のACKが2つ返ります。最初に返る!00と、それに続くLFは変更前の通信速度で返ります。
2つ目の!00とそれに続くLFは、新しく設定した通信速度で返ります。

下図は1回目のACKと2回目のACKを観察したオシロ波形です。



1回目のACKから次の新しい通信速度でのACKが返るまでの時間は実測で約1.72ミリ秒です。(環境により多少変化します。)

【補足】　本設定は、電源を切断又はハードウエアリセットをすると初期値の115200bpsに戻ります。通信速度設定を記憶することはできません。

【コマンド】 **I △ U0 :** *{LF}*

【動作】　装着されたUSBメモリーの初期化を実行する

【解説】　装着されたUSBメモリーのファイルシステムとデバイスを初期化して、使用できるようにします。USBメモリーのアクセスをはじめる前に必ず"I U0:"コマンドで初期化をします。USBメモリーを入れ替えた場合にも行う必要があります。

※"U0:"はUSBメモリードライブを示すIコマンドに対する引数です。USBH-ACS30ではUSBメモリーしか使えませんので必ず"I U0:"が初期化コマンドになります。

【使用例】　**I<sp>U0:<LF>**　USBメモリーを初期化します
!00　初期化正常完了で戻ります

【戻り値】　!00　初期化が正しく完了すると、ACKが返ります。
各種コマンドを続いて送信する際には、必ず!00が返ったことを確認してからコマンドを送信してください。
USBメモリーによってはACKが返るまでに5秒から最大50秒程度の時間がかかる場合があります。その場合でも必ずACK又はエラーコードの戻り値を待ってください。最大50秒まではACKを待つようにします。
初期化でACKが返らなかった場合、そのUSBメモリーは使用できません。!00以外の数値が返る場合には、何らかのエラーが発生しています。エラーが発生した場合には、本書の"エラーコードについて"の項をご覧下さい。

!11　USBメモリーが正しく装着されていない場合に戻ります。

【補　足】　Iコマンド送信後、エラーメッセージが返り初期化が正しく完了できない場合には、Iコマンドを再度送信して、!00が返るまで送信を繰り返してみてください。送信は最大10回程度まで繰り返し、それでも初期化できない場合にはそのUSBメモリーとの相性などが原因のことがあり、そのUSBメモリーは本機では使えません。

初期化は、コマンドの中で最も処理に時間がかかるコマンドです。早いものであれば5秒程度でACKが返りますが、容量が大きい物では最長で50秒程度時間がかかる場合もあります。そのUSBメモリーが相性問題等で使えない場合でも、USBH-ACS30はエラーコードを返します。戻り値が戻らないということは基本的にありませんので、何らかの戻り値が戻るまで処理は待機するように設計をお願い致します。不必要に大きな容量のUSBメモリーを使うと初期化に時間がかかってしまいますので、なるべく2GB～8GB程度のUSBメモリーを使用されることをお奨めします。

【コマンド】 **K △ U0 :** *{LF}*

【動作】 USBメモリーの空き容量を取得する

【解説】 USBH-ACS30に挿入されているUSBメモリーの空き容量を8バイト長の16進数値で取得します。
単位はバイトです。必ず戻り値が戻ってから次のコマンドを送信してください。
※サイズの大きなメディアの場合、取得までに時間がかかる場合があります。
※本コマンドはIコマンドによる初期化後から使用できます。

【戻り値】 下記の書式で結果が戻ります。

!00	最初のACKは空き容量計算を開始したことを通知するACKです
$*ssssssssssssssss*	*s* は空き容量の値、単位はバイトです
!00	最後のACKは空き容量取得後であることを通知するACKです

【使用例】 **K<sp>U0:<LF>**

!00	
$000000003D06D000	USBメモリーの空き容量は1,023,856,640バイトです
!00	

【コマンド】 **@** *△full_path{LF}*

【動作】 指定したディレクトリのフォルダー及びファイルリストを初期化

【引数】 *full_path* 初期化したいディレクトリをフルパスで指定します。(USBメモリーのルートディレクトリは U0: です。)

【解説】 指定したディレクトリにあるファイルとフォルダのリストを初期化します。ディレクトリの内容を閲覧する場合には"N"コマンドを使用しますが、Nコマンドを実行する前に、このコマンドを実行してリストを初期化する必要があります。
なお、Nコマンドを使う場合には、このコマンド実行後Nコマンドを実行するまでの間、別のコマンドは実行しないでください。
引数の指定は、リストを初期化したいディレクトリをフルパスで指定する必要があります。
USBメモリーのルートディレクトリは U0: です。ディレクトリの区切りは\(バックスラッシュ=0x5C)です。

【戻り値】 !00

【使用例】

@<sp>U0:<LF>	USBメモリーのルートのリストを初期化する場合
!00	
@<sp>U0:\LOG<LF>	USBメモリー内にある"LOG"フォルダ内のリストを初期化する場合
!00	

【補足】 ディレクトリの内容を閲覧するNコマンドでは、どのディレクトリの一覧を閲覧するか、という指定は行いません。@コマンドによって初期化したディレクトリがNコマンドのディレクトリ指定と同じ意味を持ちます。

【コマンド】 **N***{LF}*

【動作】 @コマンドで初期化したディレクトリの一覧を取得

【解説】 @コマンドで初期化したディレクトリにあるフォルダ名及びファイル名の一覧を取得するコマンドです。ファイル名には3文字の拡張子も含まれます。
フォルダ名又はファイル名を1つずつ表示します。表示は1つずつですので、複数のファイルやフォルダがある場合には、Nコマンドを複数回実行してます。すべてのフォルダ、ファイルを取得してこれ以上取得すべき内容がない場合"!04"を返します。
"!04"が返るまでNコマンドを実行することで、指定ディレクトリのすべてのフォルダ名、ファイル名を取得できます。

Nコマンドではフォルダ名、ファイル名の他に属性とサイズも返します。サイズは4バイト長の16進数表現です。
属性は下記の表の通りです。

【戻り値】

!00	最初にACKが返ります
NNNNN.EEE	NNNNNはファイル又はフォルダ名、EEEは拡張子(すべて大文字)
$AA	属性データ(下表参照)
$ssssssss	4バイト長のファイルサイズ　16進数表記で単位はバイト フォルダの場合には*$00000000*となります。
!00	最後にACKがつきます

*$AA*は1バイトサイズで属性データです、属性一覧に示す属性が返ります。該当するビットが1になります

ビット	5	4	3	2	1	0
属性	アーカイブ	フォルダ	ボリュームID	システム	隠しファイル	読み取り専用

属性一覧

例えば、フォルダの場合には、"010000"となりますので、戻り値のは*$10*となります。
どのビットが1になっているかを判定すると、そのファイルの属性を知ることができます。

@コマンドで指定したディレクトリが、ルートディレクトリよりも下の階層にある場合には下記のようになります。

!00	
.	ピリオド(0x2E)が1つ返ります
$10	属性は$10(10000)となり、フォルダを示します
$00000000	サイズは0になります
!00	

再度Nコマンドを送信すると下記のような文字列が返ります。

!00	
..	ピリオド(0x2E)が2つ返ります
$10	属性は$10(10000)となり、フォルダを示します
$00000000	サイズは0になります
!00	

再度Nコマンドを送信すると、そのディレクトリにあるファイル名やフォルダ名が返ります。
※ディレクトリの階層の深さに関わらずルートディレクトリよりも下にある場合には、常にこの内容になります。

【補足】 USBメモリーに存在する既存のファイル名、フォルダ名の戻り値はすべて大文字になります。

【コマンド】 **O** *△ nM>full_path{LF}*

【動作】 指定したファイルハンドルに、モードを指定してファイルをオープンする

【引数】 *n* 0～Fのファイルハンドル値を指定します、ここで指定したファイルハンドルにファイルが展開されます。
ファイルハンドルは0,1,2,3･･･E,Fの合計16個指定できます。

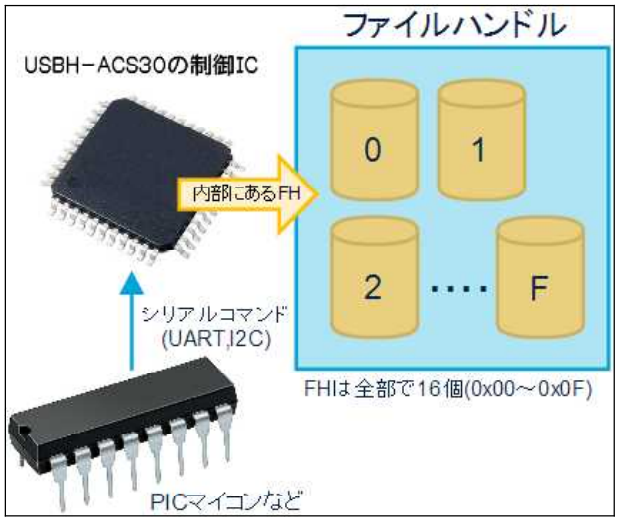
M "R" "W" "A" のいずれか1つのオープンモードを指定します
･･･"R" 読み込みモード
ファイルを読み込み専用モードで開きます。USBメモリー内にある既存のファイルを開きます。
開いたファイルからデータを読むことはできますが、データを書き込むことはできません。

･･･"W" 新規書き込みモード(ファイル新規作成)
ファイルを新規に作成し、そのファイルに新たに書き込むモードです。
既存のファイルに書き込むのではなく新しくファイルを作成するモードです。

･･･"A" 追加書き込みモード
USBメモリー内にある既存のファイルへデータを追記するための上書きモードで開きます。
指定したファイルがない場合には、新規書き込みモードになります。(Wモードと同じになります。)

full_path 開くファイルをフルパスで指定します。USBメモリーのルートは、U0:です。ディレクトリの区切りは\(バックスラッシュ=0x5C)です。フルパスで指定する時、既存のディレクトリ以外を含めるとそのファイルが作成されるディレクトリまで下層のディレクトリが自動的に作られます。

【解説】 USBH-ACS30ではファイルを扱う際には必ずファイルをファイルハンドルと呼ばれる専用のメモリ空間に開く必要があります。(本書9ページ参照)



Oコマンドは指定したファイルをファイルハンドルに開きます。ファイルハンドルはUSBH-ACS30内に0～Fまで16個あり、任意のファイルハンドルにファイルを開くことができます。一度ファイルハンドルにファイルを開いたら、次のファイル操作からはファイル名ではなくファイルハンドルを指定します。下記の例は、ファイルハンドル1にファイルを割り当てます。USBメモリーのルートディレクトリに"Folder"ディレクトリを作り、その中に"123.txt"というファイルを新規に作成しする場合です。

O<sp>1W>U0:\Folder\123.txt<LF>
!00

USBメモリーは、USBH-ACS30ではU0:ドライブと決められています。各ディレクトリの区切りは\(バックスラッシュ=0x5C)です。

もしUSBメモリー内に"Folder"というフォルダがはじめから存在している場合には、そのフォルダー内に123.txtを作ります。
もし"Folder"というフォルダが存在しない場合には、"Folder"フォルダーも作られ、その中に123.txtを作ります。

ファイルをファイルハンドルに開く際には、そのファイルをどんな目的で開くのかを指定します。これをオープンモードと呼びます。オープンモードには下記の3つがあります。

・読み込みモード　(Rモード)
・新規ファイル作成書き込みモード　(Wモード)
・既存のファイルへの上書きモード　(Aモード)

Rモードで開いたファイルには、データの追記はできません。
新規ファイル作成書き込みモード及び既存ファイルへの上書きモードで開いたファイルからはデータを読み取ることはできません。
モードを変更するには、一度ファイルハンドルを閉じてから(Cコマンド)改めてモードを設定してファイルを開き直します。
Aモードは既存のファイルにデータを上書きするモードですので、既存のファイルを指定するのが基本ですが、指定したファイルが存在しない場合には新規作成としてWモードとして開き、ファイルが新規に作られますのでご注意ください。(指定したファイルが存在しないことを通知するエラーにはなりません。)

Oコマンドを実行したら必ずACKの!00が返るのを待ってください。ACKが返った後、次のコマンドを送信してください。

【使用例】　USBメモリーのルートディレクトリにある TEST.TXT を読み込み専用モードでファイルハンドル0に開く場合

O<sp>0R>U0:\TEST.txt<LF>
!00

【補足】　USBH-ACS30では、USBメモリーに存在する既存のファイル及びフォルダを指定する場合、ファイル名及びフォルダ名の大文字、小文字は区別されません。例えば、"FOLDER"と"folder"は同じファイル名又はフォルダ名として認識されます。
USBH-ACS30で新規にファイルを作成する場合、ファイル名及び拡張子はすべて大文字になります。(使用コントローラICの仕様です。)

内蔵のRTCを有効にしている場合には、Wモード及びAモードで作成、編集したファイルにはタイムスタンプが記録されます。

Oコマンドでファイルハンドルにファイルを開いた直後は、Wモード並びにRモードの時は、カレントカーソル位置は必ず0番地となります。すなわちファイルの先頭位置にカーソルが置かれます。カレントカーソル位置を移動したい場合には、Oコマンドでファイルハンドルにファイルを開いた後、Pコマンドを使用してカーソル位置を移動させることができます。
※Pコマンドでカーソルを移動できるのは、Rモードで開いた場合のみです。

既にUSBメモリーに存在しているファイル名をWモードで開くとデータが先頭から上書きされます。(Aモードの場合にはデータの一番最後からデータが追記されます。)

【コマンド】 **C** *△ n{LF}*

【動作】　指定したファイルハンドルをクローズして、ファイルハンドルに開かれているファイルの内容をUSBメモリーに書き込む

【引数】　*fh*　　0～F　クローズしたいファイルハンドルを指定します

【解説】　ファイルハンドルを閉じます。Rモードで開いている場合にはそのままファイルハンドルを閉じます。
Wモード及びAモードで開いている場合には、データをUSBメモリー上に書き込みファイルの実体を作成し、ファイルハンドルを閉じます。ファイルは、Cコマンドでファイルハンドを閉じたときに初めてその実体がファイルとして保存されます。よって、Cコマンドでファイルハンドルを閉じる前にUSBメモリーを本体から外したり、電源を切断したり、リセットをしてしまうと、ファイルハンドルに展開されているファイルはすべて破棄されてしまいます。

【戻り値】　*!00*　　ファイルハンドルが正常に閉じると ACKの!00が返ります。ACKが返る前に電源の切断やUSBメモリーを取り外すとファイルは作成されません。

【補足】　ファイルハンドルをクローズせずにWコマンドで書き込んだデータをUSBメモリーに書き込むコマンドとしてFコマンドがあります。
Cコマンドはファイルハンドルを閉じてデータを書き込むのに対して、Fコマンドはファイルハンドルを閉じずにデータを書き込みますので、Fコマンド実行後は再び、OコマンドによってFHを開きなおさなくても、Wコマンドから始められます。Fコマンド実行後に書き込まれたデータは、すでにデータとしてUSBメモリーに書き込まれていますので、予期しない電源切断があってもFコマンド実行前までのデータは書き込まれていますのでデータの保護にもつながります。

【コマンド】 **R** *△ nF>ssssssss{LF}*

【動作】 指定したファイルハンドルのファイルの内容を指定したバイト数読み込む

【引数】 *fh* 0～F 読みたいファイルのあるファイルハンドルを指定します

F 超過文字を指定します。超過文字とは、読み取りを指定したバイト数よりも実際のデータのサイズが小さい時に、その差分を表す文字のことです。*F*で指定した半角英数字1文字で表します

ssssssss データを読み込むサイズを16進数のバイト単位で指定します
"FFFFFFFF"まで指定できます。すなわち4294967295バイトまで1回のコマンドでデータを読み込めます

【戻り値】 戻り値は、読み取ったデータの内容と、ACK及び読み取りができたデータサイズです。
書式は下記のようになります。

!00 ACKが戻ります。
d･･･d$aaaaaaaa *d* は読み込んだデータ、*s* は読み込んだデータのサイズの16進値です。
!00 最後にACKが返ります。

読み込んだデータの最後には必ず$マーク(0x24)が挿入されますので、データの最後を検出するのに使用できます。
なお、読み取り指定したサイズが、ファイルハンドルにあるデータのサイズより大きい場合には、最終データまで読み取った後、引数*F*で指定した超過文字が超過分付加されます。

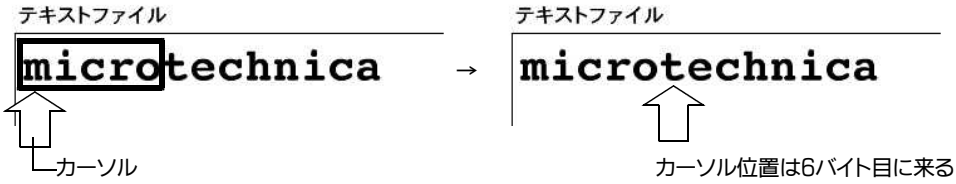
【解説】 Rコマンドは指定したファイルハンドルにRモードで開かれているファイルから指定したサイズ分データを読み取るコマンドです。
Rコマンドを使用する前に、必ず"O"コマンドにて、ファイルハンドルにファイルを読み込みモード(Rモード)で開いておく必要があります。
※Rモード以外のモード(WモードやAモード)で開いたファイルからはデータの読み込みはできません。

引数の*F*は超過文字を指定します。例えば5バイトしかないファイルに対して、7バイトの読み取りを指定した場合2バイト分が超過しています。そこでUSBH-ACS30は、超過している2バイト分を*F*で指定した文字で返します。

Rコマンドで一度ファイルを読み込むと、カレントカーソル位置(処理を開始する位置のこと)が1回前に読み込みが終わった位置の次の番地に移動しています。例えば、ファイルハンドル1に文字列 "microtechnica" が書かれた13バイトのテキストファイルがあるとして、次の例をご覧ください。

R 1Z>5
!00 ACKが返ります。
micro$00000005 データ5バイトと、読み取ったデータサイズが16進数表記で戻ります。
!00 最後にACKが戻ります。

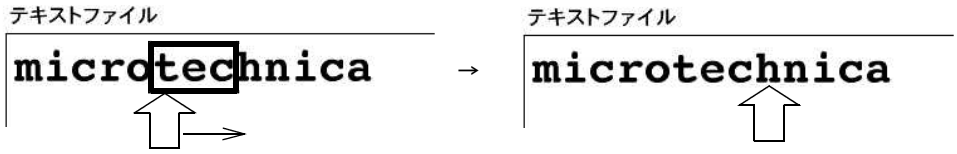
Rコマンド初回の実行ですのでカレントカーソルは先頭にあり、そこから5バイト分読み取りますので"micro"の5文字(5バイト)が返ります。(下記例では便宜上テキストファイルで文字列としてデータを表しています。)



続いて再度同じファイルハンドル1のファイルから3バイトのデータを読み取ります。

R 1Z>3
!00
tec$00000003
!00

すでにカレントカーソルは前回のRコマンドの実行によって6バイト目に来ていたため、次にRコマンドを実行すると、前回読み取りを終了した次の位置からデータが読み取られます。



カーソル位置は9バイト目に来る

続いて同じファイルハンドル1のファイルから10バイトのデータを読み取ります。

R 1Z>A
!00
hnicaZZZZZ$00000005　　9バイト目から残り5バイトのデータが返り、指定超過分の5バイトを示す超過文字Zが5つ返ります
!00

データを読み終わってもカーソルは先頭には戻らず最終位置で止まります。
先頭に戻したい時はPコマンドを使用して先頭にカーソル位置を戻します。また任意の位置にカーソルを移動させたい場合にもPコマンドを使います。

※改行を含むデータを読み出す場合
読み出すファイル内に改行コードが入っている場合でもデータはそのまま改行コードをデータとして読み出します。
Windowsで作られてたテキストファイルの場合には、改行はCRとLFの2バイトです。

【使用例】 ※ファイルハンドル1のファイルに ”12345678901234567890” というデータがある場合・・・

①ファイルハンドル1にあるファイルから、先頭番地から10バイト分のデータを読む

R 1Z>A
!00
1234567890$0000000A
!00

②先頭番地から25バイト分データを読み込む

R 1Z>19
!00
1234567890123456789ZZZZZ$00000014　　*Z*は超過文字として超過したデータ分を表示しています
!00

【補足】 Rコマンドで読み込まれるデータは、すべてファイル内のデータをそのまま読み込みます。UART通信ではフォーマットを変換したりはしていません。I2C通信の場合、ほとんどのデータはそのまま読み込まれますが、一部のデータは2バイト長になります。(詳しくは7~8ページをご参照ください。)

【コマンド】 **W** *△ n>ssssssss{LF}*

【動作】 指定したファイルハンドルにWモード又はAモードで開かれたファイルに、指定したサイズ分のデータを書き込む

【引数】 *n* 0～F データを書き込みたいファイルの開かれているファイルハンドルを指定します

ssssssss 書き込むデータサイズを16進数表記のバイト単位で指定します
"FFFFFFFF"まで指定できます。すなわち4294967295バイトまで1回のコマンドでデータを書き込むことができます

【解説】 ファイルハンドルにOコマンドのWモード又はAモードで開かれているファイルに対して、データを書き込みます。
Rモードで開かれたファイルに対しては書き込みはできません。
USBH-ACS30はWコマンドを受信後、ACKの!00を返しその後データの受信待機状態となります。このACKがUSBH-ACS30から送信されたことを確認してから、*ssssssss*で指定したサイズのデータを送信します。
ACK以外のコードが返った場合には、正しく書き込みができません。

USBH-ACS30は、Wコマンドの*ssssssss*で指定したバイト数のデータを受信するまで永久的にデータの受信を待機します。
よってWコマンドを実行する場合には、書き込むデータのサイズをあらかじめよく考慮した上で書き込むデータサイズを指定する必要があります。指定したデータサイズに達しない場合、いつまでもUSBH-ACS30はデータの到達を待ちます。(タイムアウト時間の設定はできません。)クローズコマンドを送信しても、データとして受け付けてしまいますので、ファイルハンドルを閉じることはできません。

OコマンドにてWモード(新規ファイル作成モード)でファイルをファイルハンドルに開いている場合には、必ず1バイト目(ファイルの先頭)から書き込みが開始されます。Aモード(上書きモード)でファイルをファイルハンドルに開いている場合には、必ず最終番地の次の番地から書き込みが開始されます。

ファイルにデータがすべて書き終わった場合には、必ずCコマンド又はFコマンドで実体をUSBメモリーに書き込みます。
クローズ処理を行うことで初めてファイルの実体が作成されます。(データを読み出すことのできるファイルとなります。)

【戻り値】 USBH-ACS30は、指定したサイズ分のデータを受信すると、 下記のような戻り値を返します。

$ssssssss
!00

$に続き、*ssssssss* にて実際に書き込んだデータサイズを16進数表記の文字列で表示します。
また書き込みが成功したことを示すため ACKが1回返ります。
この戻り値が返らない場合には、まだ指定したデータサイズまでデータが受信できていないことを示します。

【使用例】 ファイルハンドル1にWモードで開かれているファイルに対し、文字列 microtechnica を書き込む場合

W 1>D	書き込むサイズは16進数で表現 (0x0Dバイト = 13バイト)
!00	Wコマンドを正しく受信
microtechnica	書き込むデータを送信(Wコマンドで指定したデータサイズ分を送信します)
$0000000D	13バイトの書き込み完了の戻り値
!00	書き込み正常終了
C 1	ファイルハンドル1を閉じる
!00	ファイルハンドル1のクローズ処理完了

※上記の例では、13バイトのデータを書き込んだ後クローズ処理をしていますが、クローズ処理をしなければ、再度Wコマンドを実行してデータを書き込むことができます。適宜Fコマンドでファイルの実体をメモリメディアに保存すると予期しない電源断等でデータが失われることを防げます。

【応用】 <u>※書き込むデータに改行を挿入する場合</u>
書き込むデータが文字列で、改行を挿入したい場合には、改行コードをデータとして送信します。改行コードの挿入されたファイルを閲覧する場合、OSによって改行コードの取り扱いが異なります。下記に一覧を示します。

LF	UNIX系のシステム。Linux、Mac OS X、BeOS、Amiga、RISC OSなど
CR+LF	MS-DOS、Microsoft Windows
CR	Apple IIファミリ、Mac OS(バージョン9まで)

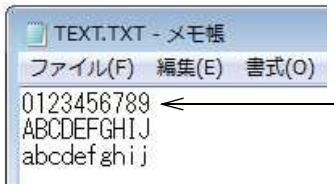
※LFは0x0A、CRは0x0Dです。

Windowsにて該当のファイルを読む場合、改行として認識させるためには、CRとLFの2バイトが挿入されてる必要があります。
CR(0x0D)しか、入っていないファイルをWindows標準のメモ帳で開くと改行が正しく反映せず、改行位置に■のような記号が入ります。
インターネットエクスプローラーや表計算ソフトのエクセルなどでは正しく改行が反映されますが、Windowsで扱うテキストデータを作る場合には改行には、CR+LFの2バイトを挿入することをお奨めします。

改行もデータサイズに含まれますので、改行が挿入されている場合には、その分Wコマンドで指定するファイルサイズを大きくする必要があります。

【使用例2】 ファイルハンドル1に新規にファイル(TEXT.TXT)を作成して改行を含むデータを作成し、それをUSBメモリーに書き込む一連の操作例

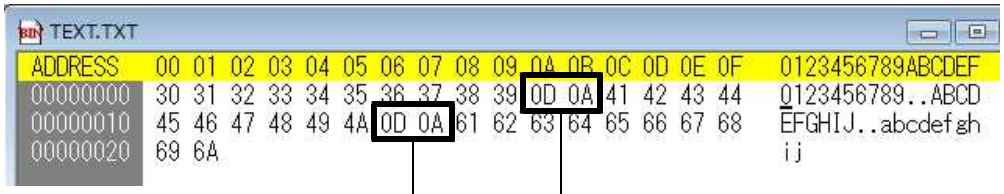
I<sp>U0:<LF>	USBメモリーを初期化
!00	初期化完了
O<sp>1W>U0:\TEST.TXT<LF>	FH1にTEST.TXTファイルをWモードで展開
!00	展開完了
W<sp>1>22<LF>	FH1に34バイト(0x22)のデータを書き込み指定
!00	完了
0123456789<CR><LF>ABCDEFGHIJ<CR><LF>abcdefghij	書き込むデータ33バイト
$00000022	33バイト書き込み完了
!00	書き込み完了ACK
C<sp>1<LF>	FH1をクローズ(実体の作成)
!00	クローズ処理完了



ここまでて作成されたファイルをWindowsのメモ帳とバイナリエディタで開くと次のようになっています。

改行が入っていることが分かります。

上のファイルをバイナリエディタで開くと下のようになっています。



改行コード0x0D(CR)と0x0A(LF)が挿入されています。

【コマンド】 **P** *Δ n>ssssssss{LF}*

【動作】 ファイルハンドルのカレントカーソルの位置を任意の位置に設定する

【引数】 *n* 0～F ファイルハンドルを指定します

ssssssss 移動したいカレントカーソルの位置を16進数表記の文字列で指定します。指定できる値は0～ファイルサイズまでです

【解説】 カレントカーソルの位置は、引数*ssssssss*で指定します。表記は16進数表記で、0～ファイルサイズまで指定できます。
カレントカーソルの位置を指定することで、Rモードで開かれたファイルの読み込み開始位置を指定することができます。
データの先頭に移動する場合には、0を指定します。

【戻り値】 *!00*

【補足】 現在のカーソル位置はYコマンドで取得できます。

【コマンド】 **Y** △ *n{LF}*

【動作】 カレントディレクトリの位置(バイト数)を取得する

【引数】 *n* 0～F ファイルハンドルを指定します

【戻り値】 ACKとカレントカーソル位置が4バイト長の16進数表記で返ります。

!00 最初にACKが返ります
$ssssssss カーソル位置が16進数表記で返ります
!00 最後にACKが返ります

【解説】 ファイルハンドルに開いているファイルについて、現在のカーソル位置を取得します。
カーソル位置のある部分からRコマンドでデータが読み取れます。

【補足】 現在のカーソル位置はPコマンドで移動できます。

【コマンド】 **D** △ *full_path{LF}*

【動作】 指定したファイル及びフォルダを削除する

【引数】 *full_path* 削除するファイルへのフルパスを指定します。最後に\を付加するとフォルダの指定になります。

【解説】 フルパスで指定したファイルを削除します。
*full_path*で指定したディレクトリの最後に\を付加するとフォルダの指定になり、フォルダが削除されます。なおフォルダを削除するには必ずフォルダ内が空である必要があります。フォルダ内にファイルやサブディレクトリがある場合、削除ができず"!03"が返ります。

【戻り値】 *!00*

【使用例】 ■USBメモリー内の"Folder"フォルダを削除する

D<sp>U0:\Folder\<LF> "Folder"フォルダを削除します、但し予めフォルダ内は
!00 空である必要があります。

【コマンド】 **?** *△ full_path{LF}*

【動作】 指定したファイル名のファイル又はフォルダ名のフォルダが、カレントディレクトリに存在するかどうかを確認する
存在しているファイル、フォルダの属性とタイムスタンプを取得する

【引数】 *full_path* 調べたいファイル名又はフォルダ名をフルパスで指定します。ワイルドカードは使用できません。
完全一致したファイル又はフォルダだけ検出します。
USBメモリーのルートは、M:です。ディレクトリの区切りは\(バックスラッシュ=0x5C)です。

【戻り値】 ■指定した文字列のファイルがカレントディレクトリに存在していた場合
ファイル、フォルダ検出のACKが返ったあと、16進数表記の4バイト長ファイルサイズと属性、タイムスタンプの日時が返ります。

!00	ファイル又はフォルダを検出したことを通知するACK
$ssssssss	16進数表記のファイルサイズ(フォルダの場合は*00000000*)
$AA	1バイトサイズの属性(下表参照)
$hh:mm:ss<sp>mm-dd-yyyy	タイムスタンプの日時、時間と日にちの間にはスペースが入ります
!00	最後にACKが返ります

*$AA*は1バイトサイズで属性データです、属性一覧に示す属性が返ります。該当するビットが1になります

ビット	5	4	3	2	1	0
属性	アーカイブ	フォルダ	ボリュームID	システム	隠しファイル	読み取り専用

属性一覧

例えば、フォルダの場合には、"010000"となりますので、戻り値のは*$10*となります。

■指定した文字列のファイルがカレントディレクトリに存在しなかった場合
!20

【コマンド】 **A** *△ full_path>new_filename{LF}*

【動作】 指定したファイル又はフォルダの名前をリネームする

【引数】 *full_path* 名前を変更したいファイル又はフォルダをフルパスで指定します。最後に\を付加するとフォルダの指定になります。
USBメモリーのルートは、U0:です。ディレクトリの区切りは\(バックスラッシュ=0x5C)です。

new_filename 新しいファイル名を指定します。3文字の拡張子も含められます。

【解説】 ファイル名やフォルダ名を任意の文字列にリネームできます。
フォルダを指定する場合には、フルパスの部分の最後に \ を付加します。

【使用例】 ■USBメモリー内"TEST.TXT"ファイルの名前を"NEWNAME.TXT"に変更する

A<sp>U0:\TEST.TXT>NEWNAME.TXT<LF>	ファイル名を変更
!00	変更完了

■USBメモリー内"123"フォルダの名前を"Folder"に変更する

A<sp>U0:\123\>Folder<LF>	フォルダ名を変更
!00	変更完了

【コマンド】 **J** *{LF}*

【動作】 USBメモリーの状態を取得する

【解説】 USBメモリーの現在の状態をビットナンバーで返します。

【引数】 なし

【戻り値】 *!00*
$*ss* *ss*は1バイト長の16進数フォーマットで返ります。下表の通り該当ビットが1にセットされます。
!00

ﾋﾞｯﾄﾅﾝﾊﾞｰ	メモリーメディアの状態
0ビット目	なし
1ビット目	なし
2ビット目	USB0(U0:)ポートのUSBメモリーマウント状態(初期化済み状態) 0･･･マウントされていない(初期化されていない) 1･･･マウントされている(初期化されている)
3ビット目	なし
4ビット目	なし
5ビット目	USB0(U0:)のUSBメモリー装着状態 0･･･装着されていない 1･･･装着されている
6ビット目	なし
7ビット目	なし

【使用例】 *!00*
$24 0x24は2進数で00100100なので、ビット2、5が1になっています。そのため次のようになります。
!00 ↓
USBメモリーが装着されてIコマンドでマウント済み

【コマンド】 **M** *ΔSΔIΔDΔLLLLLLLL{LF}*

【動作】 1つのファイルの指定した開始位置から、指定したバイト数を、別のファイルハンドルに開かれているファイルにコピーする

【引数】

S	0～F	コピー元ファイルのファイルハンドル番号、読み取り専用のRモードでファイルを開いておく必要があります。
I	0～･･･	引数*S*のファイル内のコピー開始位置を指定します。0はファイルの先頭を示します。
D	0～F	コピー先ファイルのファイルハンドル番号、Wモード又は追記の場合はAモードでオープンしておきます。
LLLLLLLL	0～･･･	コピーしたいデータのサイズを16進数で指定します。

【解説】 Rモードでオープンされたファイルハンドルのファイルを、Wモード又はAモードで開かれたファイルハンドルのファイルにコピーします。
引数Sで指定されるファイルハンドルはコピー元でデータの読み出し元になりますので、必ずRモードでオープンしておく必要があります。
引数Dで指定されるファイルハンドルはWモード又はAモードでオープンする必要がありますが次のような動作になります。

・Wモードで新規ファイル名を指定した場合: 新規にファイルが作られ、コピーされてデータがファイルの先頭から書き込まれます。
・Wモードで既存ファイル名を指定した場合: 既存のファイルの先頭からコピーしたデータが上書きされます。既存データは消えます。
・Aモードで新規ファイル名を指定した場合: 新規にファイルが作られ、コピーされてデータがファイルの先頭から書き込まれます。
・Aモードで既存ファイル名を指定した場合: 既存のファイルの最後尾からコピーしたデータが追記されます。既存データは残ります。

注意が必要なのは、引数*S*で指定するファイルハンドルのファイルがRモード以外でオープンされている場合、このコマンドを実行するとファイルの構造が壊れることがあることです。コピー元ファイルは常に読み取りモード(R)でしか開けませんので、それ以外のモードで開いているファイルハンドルを指定することは絶対にしないでください。
さらに、コピー先のファイルはWモード又はAモードでオープンされている場合に限り使えます。Rモードでオープンされているファイルハンドルをコピー先に指定(引数*D*)するとファイルの構造が壊れることがありますので、注意してください。

引数*D*で指定したコピー先ファイルはオープンモードが、WモードかAモードによって動作が変わりますので上記を参考の上注意して指定してください。

【戻り値】 *!00*

$*xxxxxxxx*	コピーが実行されたデータのサイズが16進数で返ります。
!00	ACKにてコピー完了が通知されます。

【コマンド】 **E △ U0：***>ssssssss{LF}*

【動作】 USBメモリーの書き込み、読み込み転送スピードをテストする

【引数】

ssssssss	テスト転送するデータのサイズを4バイト長の16進数表記で指定します。 値は1024の倍数にします （例:100MBの場合は6400000を指定します）

【戻り値】 指定したデータサイズの読み込み時間（ミリ秒）と、書き込み時間（ミリ秒）を返します

!00	読み書きテスト開始を示すACKが返ります。テストには一定の時間がかかります。
$aaaaaaaa	データ書き込み時間が16進数表記で返ります。単位はミリ秒です。
$bbbbbbbb	データ読み込み時間が16進数表記で返ります。単位はミリ秒です。
!00	テストが正常に完了したことを示すACKが返ります。

【解説】 USBメモリーの読み書きに要する時間を計測するコマンドです。このコマンド実行には最低2つのファイルハンドルが空いている必要があります。
このコマンド実行には時間がかかります。時間はUSBメモリーのサイズとテストに使用するデータのサイズに依存します。
このコマンドの結果は実際にファイルを書き込み、読み込みすることでその時間を実測して結果を表示します。表示時間はあくまでも実験や評価の参考としてお使いください。
このコマンドを実行する前にUSBメモリーをフルフォーマットしておくことをお奨めします。

【使用例】

E<sp>U0:>6400000<LF>	U0:のUSBメモリに100MBのデータを転送してテストする
!00	テスト開始
$00019194	書き込みに要した時間　102804ミリ秒
$0001C2E1	読み込みに要した時間　115245ミリ秒
!00	

上記の場合、書き込み速度が何bpsなのかを求めたい場合には次のように計算します。
100MB/102804ミリ秒=972.7247967kB/sec　約973kB/secとします。
973kB/sec = ((973kB×8)/1)=7.784Mbps

※100MB = 1024 × 1024 × 100 = 104,857,600バイト → 0x6400000

【コマンド】 **Q △ CONFIRM △ FORMAT △ U0:** *{LF}*

【動作】 装着されたUSBメモリーをクイックフォーマットする

【解説】 USBメモリーのクイックフォーマットを行います。ローレベルフォーマットはできません。
ファイルシステムのファイル管理情報の保存領域のみを消去します。フォーマットすると、USBメモリー内のすべてのデータは削除されます。フォーマットには時間がかかる場合があります。

【戻り値】 *!00* 最初のACKはクイックフォーマットの開始を通知するものです
!00 2回目のACKはフォーマットが完了したことを通知するものです

2回目のACKが返るまでの間がクイックフォーマットしている時間です。
2回目のACKが返るまではメモリーカードの抜き差しをしないようご注意下さい。

【補足】 クイックフォーマットに要する時間はメモリーカードの容量や、メーカー等によってかなり変動します。
実際に使用するメモリーカードを使用して実測されることをお奨めします。長い場合3分程度かかる場合もあります。

クイックフォーマットだけする場合でも最初にIコマンドによるメモリメディアの初期化が必要です。

2回目のACKが返る前(クイックフォーマット中)に、USBメモリーを外したり、本体の電源を切断するとファイルシステムが破損しパソコンでフォーマットするまで、そのUSBメモリーは使えなくなります。

エラーコード一覧

USBH-ACS30は送信したコマンドに対してエラーが発生するとエラーを返します。下表の値は、"!"マークの後ろにつく値を示しています。

※エラーコードはあくまでもエラーの目安であり、時に不正確な(下表に記載でない内容)エラーが返ることがあります。

ｴﾗｰｺｰﾄﾞ	エラーの内容
00	エラーなし(ACK)
01	理解できないコマンドです
02	引数が不正です
03	動作及び操作に失敗しました 書き込み禁止されているメディアへの書き込み
04	ファイル、フォルダリストが最後に到達
10	メディアが初期化されていません
11	初期化に失敗しました
12	ストレージメディアの空き容量が不足しています
20	ファイル、フォルダが存在しません
21	ファイルオープンに失敗しました
22	シークはRモードで開いたファイルのみ可能です
23	シークサイズの上限はファイルサイズまでです
24	ファイル名は0にできません
25	ファイル名に不正な文字が含まれています
26	ファイル、フォルダ名はすでに存在しています
30	不正なハンドルです
31	ハンドルソースを開けません
32	ハンドルの指定先が開けません
33	ハンドルソースはRモードでのオープンを要求します
34	ハンドルソースはWモード又はAモードでのオープンを要求します
35	空きハンドルはこれ以上ありません
36	ハンドルを開けません
37	ハンドルはすでに使用されています
38	オープンファイルモードが不正です
39	ハンドルはWモード又はAモードを要求しています
3A	ハンドルはRモードを要求しています
40	システムはビジーです
41	このコマンドはSPI通信の時のみサポートされます
FF	ブートローダインディケーションコード

■エラー発生時の対応方法

USBH-ACS30では、ほとんどのコマンドで正しく処理が完了したり、コマンドを正しく受信すると、ACKとして!00が返るような仕組みになっています。詳しくは各コマンドの詳細に記述されていますが、多くの場合、USBH-ACS30に接続するシリアル通信側の機器は、この!00を待ってから次の操作を行う必要があり、!00のACKを待たないで、又はACKの受信をしないで次の操作を行うことは、様々な問題に繋がります。

USBH-ACS30が!00以外のエラーコードを返す場合には何らかのトラブルが発生していることを示しており、エラーコードが返ってくる以上正常な動作を期待できません。よって、USBH-ACS30と接続する機器はこのエラーコードを受信したら、処理を一度停止し原因を解決しなくてはなりません。原因は特定が容易なものから特定ができない難しいものまで含まれています。多くの場合には次のような方法でエラーに対処することが望ましいと考えられます。

①エラーコード表からエラーを特定する

エラーコード表を参照して原因の特定を試みます。原因が分かれば、それに対応した方法でエラーを回避するようにします。

②エラーが解決しない場合

エラーは各機器がすべて正常に動作していても、タイミングのずれや、信号に物理的な変動などの不確定要因が生じていたり、外部からのノイズなど、様々な要因で発生することがあります。それらのエラーであった場合には、同じルーチンを繰り返し実行することで、正常に動作することがあります。エラーが発生している場合で、エラーの原因が特定できないような場合には、再度同じコマンドを発行することで問題が回避できるかもしれません。少なくとも10回程度は同じコマンドを発行してみてACKが返ってくることがあるかどうかについて確認してみてください。

③本体をハードウエアリセットする

エラーが発生し続けている場合には、一度USBH-ACS30本体のハードウエアリセットをすると問題が回避できるかもしれません

④電源を再起動する

ハードウエアリセットをかけても、本体のCPUはリセットされますが、取り付けたUSBメモリーはリセットされていません。電源を再起動することでUSBメモリーもリセットされますので、③試行後でもエラーが続く場合には一度本体を再起動することが有用である場合があります。

⑤USBメモリーのフォーマットをしてみる

USBメモリーへの書き込みや読み込みは、FATというファイルシステム仕様に基づいて行われています。ファイルシステムに問題が生じていたり、正しくフォーマットができていないメモリーデバイスを使うと、エラーが発生したり予期しない問題が発生することになります。一度WindowsでUSBメモリーをフォーマットし直してお試し下さい。なお、フォーマットを行う場合にはクイックフォーマットは使わないでフルフォーマットをお試し下さい。

⑥相性問題を疑ってみる

パソコンなどで使用されるUSBメモリーは汎用的な製品であり、ある規格に即して設計、製造されています。よって一般的にはその規格を満たす機器同士であれば正常に使えるはずです。

しかし、個別の機器自体は正常に動作しており、規格に即した仕様になっているにも関わらず、組み合わせて使用すると正しく動作しなかったり動作が不安定だったり・・という問題が発生することがあります。これを一般的には相性問題と呼んでいますが、規格がオープンで広く普及した規格では相性問題がより起こりやすいと言えます。

相性問題が発生する理由は、定められた規格の仕様に一定の余裕が持たせてあるためです。例えば時間的なタイミングにしても、最小〇〇μ秒～最大〇〇μ秒まで・・・というように仕様に幅があるものです。そのため仕様範囲内であっても、タイミングのずれが重なると、許容範囲を逸脱してしまったり、動作に問題が生じたりすることがあります。

また、高速で動作する機器の場合には、分布定数回路が形成されて、波形が歪んだり、波形の立ち上がりや立ち下がりになまりが生じて、その結果、問題が発生するということもあります。

相性問題は、汎用的な規格を用いる場合にはなかなか避けられない問題で、その問題が発生した場合には解決の方法が少ないのも事実です。そういった場合には機器を交換するしかありません。

もし、動作がどうしてもうまくいかなかったり、不規則な問題が生じる場合には相性問題を疑って頂き、使用するUSBメモリーを変えてお試し頂くなどの対応が必要かと思います。

ファームウエアのアップデートの方法

本製品に使用されているALFAT SoCのファームウエアは既知の問題点や新機能の追加、相性問題等の解決などを目的としてアップデートされることがあります。アップデートを行う場合には、USBH-ACS30のUART通信端子(TXおよびRX)をRS232Cレベル変換IC等を介してパソコンと通信させられる環境が必要です。アップデートはWindowsパソコンから専用ツールにてCOMポート経由で行われます。また、ALFAT SoCは動作モードをBootloaderモード(手順5参照)にしておく必要があります。次の方法でアップデートしてください。

ファームウエアに修正が加えられると、アップデートファイルを当方のサイトよりリリース致します。アップデートを行うことで既知の問題が修正されたり、新しい機能が追加される場合があります。後から問題を修復させられるよう、ファームウエアのアップデートができるようにシステムを開発して頂く必要があります。ファームウエアのアップデートを行うには最低次の要件を満たせるハードウエア環境が必要になります。

・Mode1ピン、Mode2ピンが設定できる。
・UARTピン(TX.RX)とGNDが外部に取り出せる。
・UART経由でパソコンのUSBポート又はRS232Cポートと接続できる
・Windowsパソコンでアップデートソフトが動かせる

1 当方のFAQサイトにて新しいファームウエアがリリースされていないかご確認ください。現在のALFAT SoCに書き込まれているファームウエアバージョンは"V"コマンドにて確認できます。
http://www.microtechnica.tv/faq/faq.cgi?mode=kate&kate=ALFAT

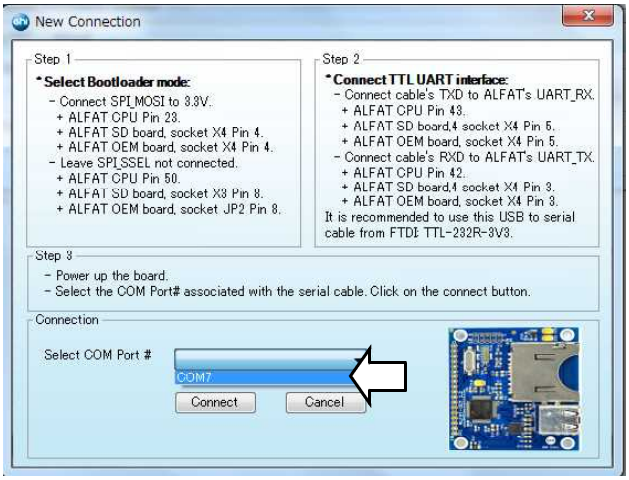
2 上記のFAQサイトからアップデートツール"Firmware Updater"をダウンロードしてください。zip形式で圧縮されていますので、解凍しておいください。

3 上記サイトから最新版のファームウエアのダウンロードしてください。ファームウエアファイル本体の拡張子はGHIです。

4 パソコンとUSBH-ACS30を接続した状態で手順2でダウンロードしたzipファイルを解凍すると展開される"ALFAT_Updater_v100.exe"を実行してください。

5 USBH-ACS30をブートローダーモードで起動します。
ブートローダーモードは、USBH-ACS30のMode1及びMode2を次の状態にして本体に5Vの電源を給電して起動します。
・Mode1ピン(5ピン)→Hレベル
・Mode2ピン(8ピン)→Lレベル
上記状態に設定した後、3.3Vを給電して起動してください。

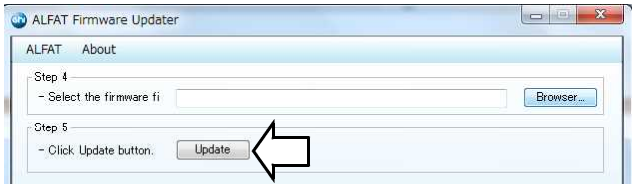
6 "Serial COM Port #"のプルダウンから、USBH-ACS30が接続されているCOMポートを選択してください。



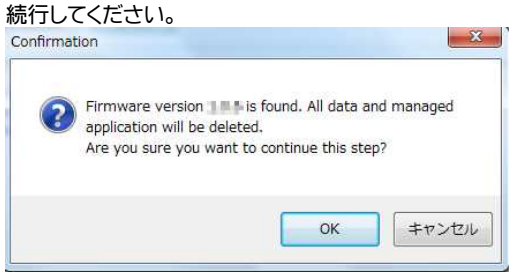
7 "Connect"ボタンをクリックして接続します。
接続して通信ができていることを確認します。メニューバーの"ALFAT"から"Check Version"をクリックします。
正しく通信できていると、"Firmware version x.x.x"と現在のファームウエアのバージョンが表示されます。
表示されない場合には通信が正しくできていませんので、正しくパソコンと通信のできる環境にしてください。

8 "Browse"ボタンをクリックして最新のファームウエアを読み込みます。

9 書き込みを開始しますので、"Update"ボタンをクリックします。



下図のようなメッセージが表示された場合には、OKボタンを押して続行してください。



アップデート中はUSBH-ACS30の電源やパソコンの電源を切断しないでください。また接続を切断しないでください。
プログレスバーが伸びて進捗状況を表示します。

アップデートが完了したら、バージョンを確認して本体を再起動して完了です。

使用上の注意

USBH-ACS30を使用するに際して、必ず下記の注意事項をお守りください。

①USBH-ACS30を使用するに際し、当方は明示的及び潜在的な使用したことによる危険性や、不確実性については予見することができません。使用する際には、お客様の責任においてこの製品を正しくお使いいただけますようお願い致します。

②USBH-ACS30は、USBメモリーに対してデータを記録したり、データを読み込むことのできる製品ですが、信頼性は一般的な使用の範囲に限定されます。本製品を宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性を要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途向けには使用できません。

③USBH-ACS30では様々な外的要因等によって、データを正しく書き込めなかったり、読み込んだデータに誤りがある場合などがあります。本製品を使用することによって生じた、もしくはこれに関連するいかなる直接・間接損害、懲罰的損害、その他データの破損や消失等を含むいかなる損害、損失についても、当方では一切責任を負いかねます。あらかじめご理解とご了承頂けますようお願い致します。

④本製品を使用した製品等を製造させる場合には、様々なフェイルセーフ機能(安全設計)を施して頂き、十分に機器のテストをした上で運用されますようお願い致します。また、データの損失や予期しない事態に備え、データのバックアップを行って頂けますようお願い致します。

⑤本製品は日本国内での使用、消費を前提として輸入された製品です。本製品の技術は米国で開発されたものであり、輸入に際しては最終仕向国を日本としています。よって本製品を第三国に輸出することは想定されておらず、当方では輸出に関わるあらゆる責任を負いません。また輸出に際して必要な各種証明書等の発行はしておりません。
RoHS指令や環境負荷物質等の調査書や報告書などの提出はしておりません。当方からお示しできる資料は本マニュアルと当方のFAQサイトに記載されているドキュメントがすべてです。

⑥将来のファームウエアアップデートに備え、必ず機器に組み込んだ後もファームウエアがアップデートできるようにシステムを設計して頂く必要があります。ファームウエアのアップデートができないと、バグが見つかった場合でもその問題を修正できません。
ファームウエアのアップデートはパソコンとUSBH-ACS30がUART通信で接続できる必要があります。

⑦動作中に本体動作が不安定になった場合など必要に応じてリセットができるようにしておく必要があります。リセットは、リセットピンをLレベルにすることで行います。何らかの方法で動作が不安定になった時、リセットできる仕組みをご用意下さい。

製品の技術的なサポートについて

本製品の技術的なサポートは製品の開発元、米GHI Electronics社が直接行います。技術的なサポートが必要な場合には、開発元へ直接ご連絡頂きます。当方での技術的なサポートは致しておりません。

技術サポートはすべて英文となります。当方(日本)での技術サポートは行っておりません。あらかじめご了承頂けますようお願い申し上げます。なお、日本語マニュアル及びFAQにつきましては、当方のサイトより最新の情報をご提供致します。

サポートは、本製品に使われているCPUの製品名、"ALFAT SoC"のサポートとして開発元が行っています。開発元ではサポートフォーラムを開設しています。まずは本マニュアルや英語版のALFAT SoCデータシートをお読みの上、解決方法を探して下さい。

もし解決方法が見つからない、疑問点がある、バグが疑われる現象がある場合などはサポートフォーラムにその内容を詳しく投稿してください。サポートフォーラムは世界中にユーザーが閲覧しており、メーカー技術者もサポートを行っており、適切な回答が期待できます。

なお、サポートはいずれも技術者専門となっており、技術的なご質問をするためのものとなっております。マニュアルに既に記載されている内容などについては回答が得られない場合がございます。

https://www.ghielectronics.com/community/forum/board?id=27

投稿には予めユーザー登録が必要です。ユーザー登録をする場合には上記サイトの右上にある"Log In"のリンクをクリックした後表示されるログイン画面で"I don't have an account."をクリックすると登録が可能です。

主な仕様

電源電圧:	本体DC3.3V(±3%以内)
	USBメモリーDC5.0V
Vbatピン:	DC1.65V～3.6V
消費電流:	通常動作時 平均約30mA
	スタンバイモード時 25μA(max)
	ストップモード時 200uA(max)
Vbatピン消費電流:	平均1.32μA (3.3V時)
動作環境:	-10℃～70℃ (動作保証範囲)
対応USBメモリー規格:	USB2.0又はUSB3.0規格
対応メモリーサイズ:	512MB～64GB
	但し実用範囲として2GB～8GBを推奨
対応ファイルシステム:	FAT16 、FAT32
	※LFNライセンス取得済み
シリアル通信方式:	非同期式シリアル通信UART
	同期式シリアル通信I2C
CPU:	ALFAT SoC (GHI Technology社製)
開発元:	米GHI Electronics社
生産国:	日本、中国

※その他の電気的特性は本書5ページを参照下さい。

マイクロテクニカ
〒158-0094 東京都世田谷区玉川1-3-10
(C)2014 Microtechnica All rights reserved